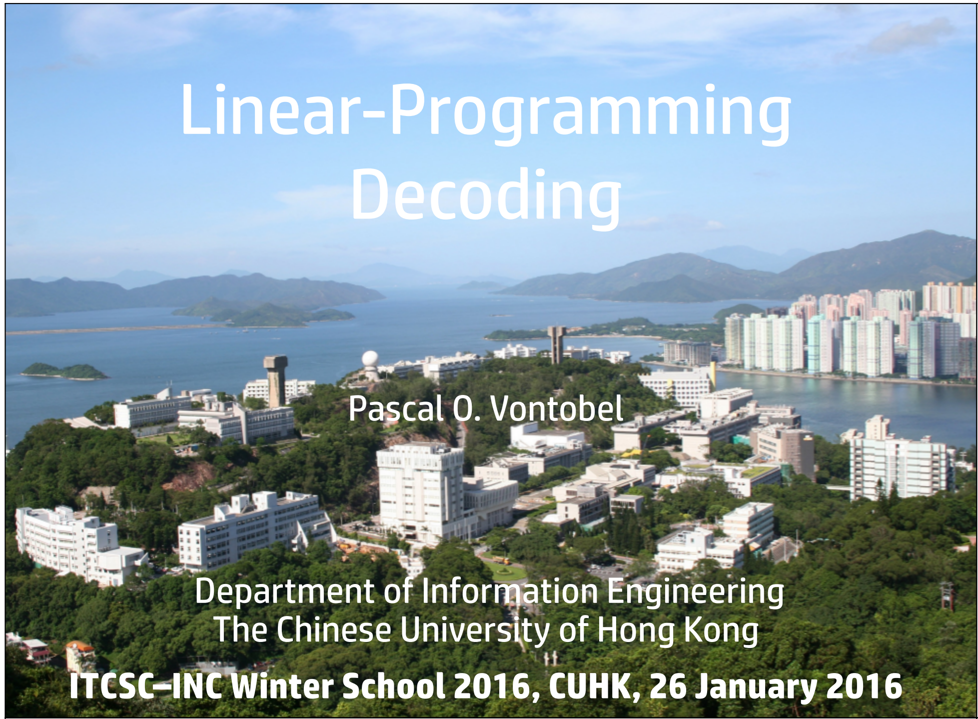


Part 1

Introduction to Coding Theory

Outline of all Parts

- Part 1: Introduction to coding theory
- Part 2: Some important concepts from coding theory
- Part 3: Maximum-likelihood decoding
- Part 4: Linear-programming decoding
- Part 5: Graphical representation of codes
- Part 6: Graph-cover decoding

Outline of Part 1

- What is channel coding about?
- Applications of coding theory
- A simple code example
- “Driving forces” for coding schemes
- Simplified communication model
- Connections to other fields

What is channel coding about?

What is Channel Coding about?

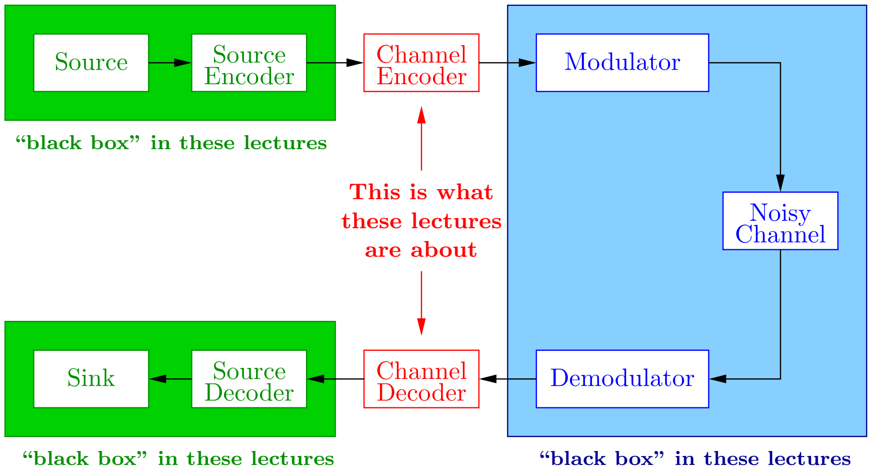
Using channel codes, we can make data transmission / storage more reliable.

The main idea is to add redundancy,
i.e., we transmit more than strictly required.

This redundancy allows us to correct, up to some limits, errors that happen during transmission / storage.

What is Channel Coding about?

Block diagram of a digital data transmission / storage system



What is Channel Coding about?

Information theory tells us what the largest possible transmission rates are (bits / channel use) for a given channel under the assumption of using the best possible encoder and decoder.

However, information theory gives us "only" the existence of such encoders and decoders.

Coding theory is about finding such encoding and decoding schemes.
Efficiency and practicality of these schemes is important!

Applications of coding theory

Applications of Coding Theory

1. Wireless communication
   Earth to satellite and satellite to earth.
   Mobile phone to base station and base station to mobile phone.
2. Wire-based communication
   Modems, DSL, fiber-optic communication, etc.
3. Optical recording
   CDs, DVDs, BluRay discs, etc.
4. Magnetic recording
   Tapes, hard disks, etc.
5. Computer memories
   Especially in high-relilability computing systems (banking, etc.)
6. Non-volatile memory
   Flash memory, phase-change memory, etc.

Applications of Coding Theory

7. ISBN (International Standard Book Number): ISBN-10 and ISBN-13
   Among the few codes designed for encoding and decoding by humans.
   Can detect a few errors that humans typically make when copying numbers.
   (More details later.)

   ISBN 0-471-06259-6

   ISBN 978-0521-55374-2

8. QR code (Quick Response code)

   (QR code examples from wikipedia)

   QR codes are based on BCH and Reed–Solomon codes.

Applications of Coding Theory

9. Hardware design
   Sometimes a wire connection pattern needs to satisfy some constraints.
   Problem can be formulated as finding / designing a code with certain properties.
10. Morse code (developed in the 1830s)
    Not really a channel code. More like a **source code** or a **modulation code**.
11. Etc.

ISBN (International Standard Book Number)

ISBN-10

ISBN-10 codeword example:

ISBN 0-471-06259-6

$$\Rightarrow \mathbf{x} = (0,\ 4,\ 7,\ 1,\ 0,\ 6,\ 2,\ 5,\ 9,\ 6).$$

(The vector $\mathbf{x}$ is a row vector of length 10.)

Definition of ISBN-10: The vector $\mathbf{x}$ is a valid ISBN-10 codeword if

$$\mathbf{H} \cdot \mathbf{x}^{\mathsf{T}} = \mathbf{0}^{\mathsf{T}} \pmod{11},$$

where

$$\mathbf{H} \triangleq (1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8,\ 9,\ 10).$$

(The matrix $\mathbf{H}$ has size 1×10.)

Comments on ISBNs

There are two ISBN standards:

- **ISBN-10** (old)
- **ISBN-13** (new)

ISBN-10

ISBN-10 codeword example:

$$\mathbf{x} = (0,\ 4,\ 7,\ 1,\ 0,\ 6,\ 2,\ 5,\ 9,\ 6).$$

(The vector $\mathbf{x}$ is a row vector of length 10.)

Verification that $\mathbf{x}$ is an ISBN-10 codeword:

$$\begin{aligned}
\mathbf{H} \cdot \mathbf{x}^{\mathsf{T}} &= (1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8,\ 9,\ 10) \cdot \\
&\quad (0,\ 4,\ 7,\ 1,\ 0,\ 6,\ 2,\ 5,\ 9,\ \ 6)^{\mathsf{T}} \\
&= 1 \cdot 0 + 2 \cdot 4 + 3 \cdot 7 + 4 \cdot 1 + 5 \cdot 0 + 6 \cdot 6 + 7 \cdot 2 + 8 \cdot 5 + 9 \cdot 9 + 10 \cdot 6 \\
&= 0 + 8 + 21 + 4 + 0 + 36 + 14 + 40 + 81 + 60 \\
&= 0 + 8 + 10 + 4 + 0 + 3 + 3 + 7 + 4 + 5 \\
&= 44 \\
&= 0 \pmod{11}.
\end{aligned}$$

Comments on ISBN-10

- First 9 symbols are information symbols ("payload").
  Information symbols are elements of $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$.
- The last symbol is a check symbol.
  The check symbol is an element of $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, \mathrm{X}\}$.
  (Here, "X" is used to represent 10.)
- Basically, information symbols could also take on the value X; however, this is not used.
- The matrix $\mathbf{H}$ is called a parity-check matrix.

Properties of ISBN-10

- Can detect any one-symbol error.
- Can detect any pair of symbol switches.
- Designed for a "human channel" (non-typical in that sense).
- Cannot correct one-symbol errors.

Proof: Omitted.

Modified ISBN-10

Question:

How can we modify $\mathbf{H}$ such that we can correct one-symbol errors?

Answer:

A possibility is given by the parity-check matrix

$$\mathbf{H}' \triangleq \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 1^2 & 2^2 & 3^2 & 4^2 & 5^2 & 6^2 & 7^2 & 8^2 & 9^2 & 10^2 \end{pmatrix}$$
$$= \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 1 & 4 & 9 & 5 & 3 & 3 & 5 & 9 & 4 & 1 \end{pmatrix} \pmod{11}.$$

Modified ISBN-10

Definition of Modified ISBN-10: The vector $\mathbf{x}$ is a valid modified ISBN-10 codeword if

$$\mathbf{H}' \cdot \mathbf{x}^{\mathsf{T}} = \mathbf{0}^{\mathsf{T}} \pmod{11},$$

where $$\mathbf{H}' \triangleq \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 1^2 & 2^2 & 3^2 & 4^2 & 5^2 & 6^2 & 7^2 & 8^2 & 9^2 & 10^2 \end{pmatrix}.$$

Properties:

- 8 information symbols
- 2 check symbols
- Can correct one-symbol errors.

Price that we pay for this enhanced coding scheme: reduction of rate from $\frac{9}{10}$ to $\frac{8}{10}$.

Here: $\mathrm{rate} = \frac{\#\text{information symbols}}{\text{codeword length}}$.

Central Theme of Coding Theory

Coding theory is about finding codes

with the best eror correction capability

for a given length and rate.

"Driving forces" for coding schemes

ISBN-13

Definition of ISBN-13:

The vector $\mathbf{x} = (x_1, \ldots, x_{13})$ is a valid ISBN-13 codeword if

$$\mathbf{H} \cdot \mathbf{x}^{\mathsf{T}} = \mathbf{0}^{\mathsf{T}} \pmod{10},$$

where

$$\mathbf{H} \triangleq (1,\ 3,\ 1,\ 3,\ 1,\ 3,\ 1,\ 3,\ 1,\ 3,\ 1,\ 3,\ 1).$$

(The matrix $\mathbf{H}$ has size 1×13.)

"Driving Forces" for Coding Schemes

1. New channels

   e.g., flash memories, phase-change memories, etc.

2. More efficient hardware

   which allows more sophisticated algorithms, etc.

3. New mathematical insights

4. New regulations

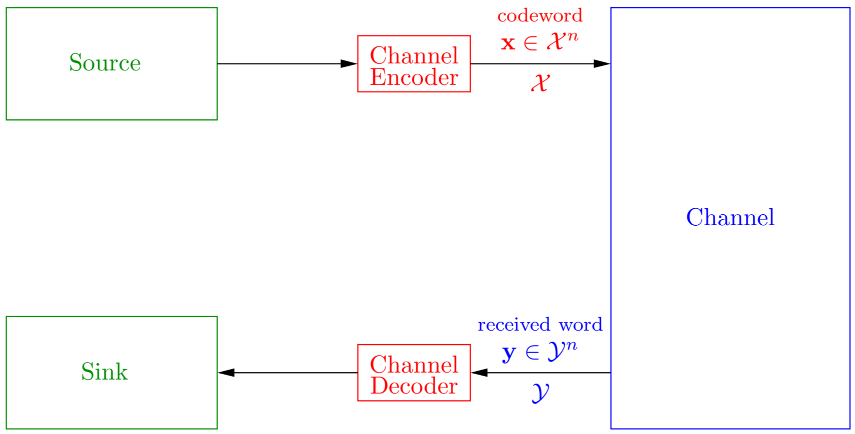
   e.g., new frequencies that become available for public wireless comm., etc.

5. Etc.

A simplified data communication model

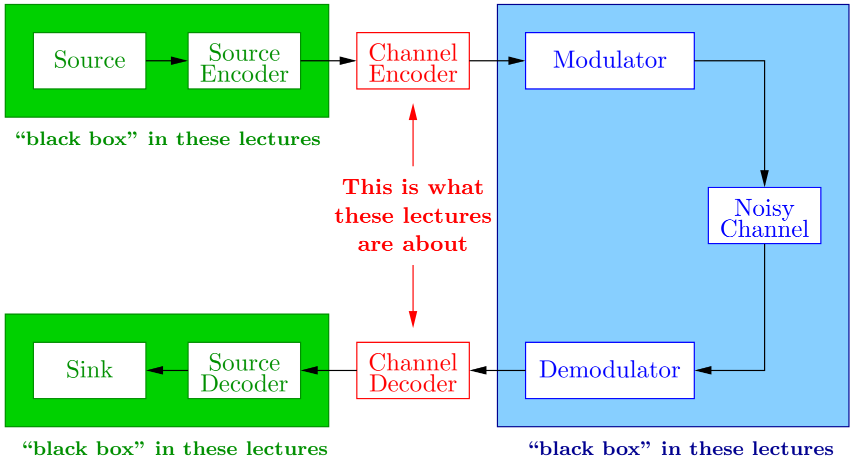
Simplified Data Communication Model

Block diagram of a digital data transmission / storage system



Data Communication Model

Block diagram of a digital data transmission / storage system



Some Basic Definitions

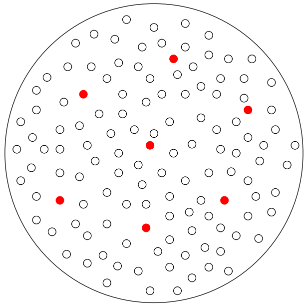
- A codeword $\mathbf{x}$ is a vector (usually a row vector) over the alphabet $\mathcal{X}$ and of length n, i.e., $\mathbf{x} \in \mathcal{X}^n$.
- The main idea of channel coding is that the set of codewords is restricted to some subset of $\mathcal{X}^n$.
- This subset is called code or codebook.
- Here we use the letter $\mathbb{C}$ to denote a code. With this: $\mathbb{C} \subseteq \mathcal{X}^n$.

  Important: $\mathbb{C}$ has nothing to do with complex numbers!

An Observation

In order to be able to correct errors,

the elements of $\mathbb{C}$ should be as far apart as possible.



$\mathcal{X}^n$: all dots

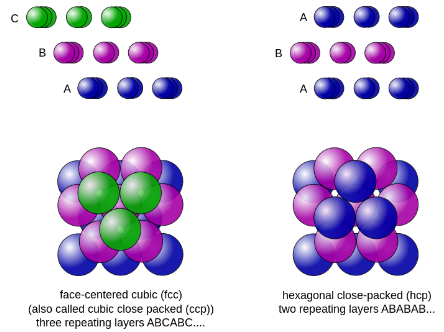
$\mathbb{C}$: dark red dots

Note:

- If $\mathcal{X}$ is discrete then $\mathcal{X}^n$ is a discrete space.
- If $\mathcal{X}$ is continuous then $\mathcal{X}^n$ is a continuous space.

Sphere-Packing Problem for $\mathcal{X} = \mathbb{R}$

Kepler was the first person to consider the sphere-packing problem for $\mathbb{R}^3$.

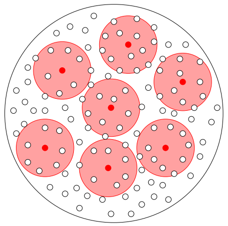


(from wikipedia)

- **Kepler's conjecture (1611)**: in three-dimensional Euclidean space, no arrangement of equally sized spheres filling space has a greater average density than that of the cubic close packing (face-centered cubic) and hexagonal close packing arrangements. The density of these arrangements is around 74.04%.
- **Proved by Thomas Hales in 1998 / 2014.**

A Simplified Data Communication Model

Assuming $\mathcal{Y} = \mathcal{X}$, we can draw decision regions in $\mathcal{Y}^n = \mathcal{X}^n$ for every codeword $\mathbf{x} \in \mathbb{C}$.



$\mathcal{X}^n$: all dots

$\mathbb{C}$: dark red dots

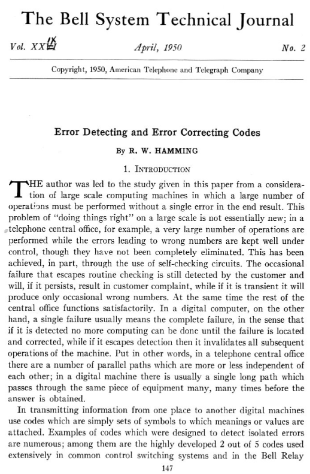
Decoder:

- If $\mathbf{y}$ is in the decision region of codeword $\mathbf{x}'$ then the decoder produces the estimate $\hat{\mathbf{x}} = \mathbf{x}'$.
- If $\mathbf{y}$ is in no decision region, then the decoder declares failure.

Note:

If the decision regions are "spheres," packing as many "spheres" as possible in $\mathcal{X}^n$ is called the sphere-packing problem.

The First Algebraic Coding Paper



The Bell System Technical Journal

Vol. XXIX *April, 1950* *No. 2*

Copyright, 1950, American Telephone and Telegraph Company

Error Detecting and Error Correcting Codes

By R. W. HAMMING

1. Introduction

THE author was led to the study given in this paper from a consideration of large scale computing machines in which a large number of operations must be performed without a single error in the end result. This problem of "doing things right" on a large scale is not essentially new; in a telephone central office, for example, a very large number of operations are performed while the errors leading to wrong numbers are kept well under control, though they have not been completely eliminated. This has been achieved, in part, through the use of self-checking circuits. The occasional failure that escapes routine checking is still detected by the customer and will, if it persists, result in customer complaint, while if it is transient it will produce only occasional wrong numbers. At the same time the rest of the central office functions satisfactorily. In a digital computer, on the other hand, a single failure usually means the complete failure, in the sense that if it is detected no more computing can be done until the failure is located and corrected, while if it escapes detection then it invalidates all subsequent operations of the machine. Put in other words, in a telephone central office there are a number of parallel paths which are more or less independent of each other; in a digital machine there is usually a single long path which passes through the same piece of equipment many, many times before the answer is obtained.

In transmitting information from one place to another digital machines use codes which are simply sets of symbols to which meanings or values are attached. Examples of codes which were designed to detect isolated errors are numerous; among them are the highly developed 2 out of 5 codes used extensively in common control switching systems and in the Bell Relay

147

Hamming wrote the first algebraic coding paper:

R. W. Hamming, "Error detecting and error correcting codes," *Bell System Technical Journal*, vol. 29, pp. 147–160, April 1950.

Connections to other fields

Part 2

Some important concepts from coding theory

Connections to Other Fields

1. Combinatorics

   designs, Hadamard matrices, difference sets, etc.

2. Algebra
3. Geometry
4. Group theory

   Golay code has lots of symmetries, the classification of finite simple groups would not have been completed without coding theory, etc.

5. Theoretical computer science

   expander graphs, derandomization, probabilistically checkable proofs, etc.

6. Physics

   spin glass models, Ising model, etc.

Outline of Part 2

- Simplified setup
- Some simple encoders, generator matrix, parity-check matrix
- Channel models
- Error detection and error correction
- Information theory
- Some notation
- Hamming distance and weight

Simplified setup

Simplified Setup

In these lectures, we will mostly consider the following setup.

Encoder mapping:	$E : \mathcal{U}^k \to \mathcal{X}^n$
Decoder mapping:*	$D : \mathcal{Y}^n \to \hat{\mathcal{X}}^n$
	$D : \mathcal{Y}^n \to \hat{\mathcal{U}}^k$
Code (also called codebook):	$\mathbb{C} \triangleq \{E(\mathbf{u}) \in \mathcal{X}^n \mid \mathbf{u} \in \mathcal{U}^k\}$
Code length:	n
Code size:	$\lvert\mathbb{C}\rvert$

Note: the encoder is usually chosen to be an **injective** mapping, i.e., distinct $\mathbf{u}$'s are mapped to distinct $\mathbf{x}$'s. With that, $|\mathbb{C}| = |\mathcal{U}^k| = |\mathcal{U}|^k$.

* Usually clear from the context which one of these two decoder mappings is considered.

Simplified Setup

In these lectures, we will mostly consider the following setup.

Information word:	$\mathbf{u} = (u_1, \ldots, u_k) \in \mathcal{U}^k$
Codeword (sent word):	$\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{C} \subseteq \mathcal{X}^n$
Received word:	$\mathbf{y} = (y_1, \ldots, y_n) \in \mathcal{Y}^n$
Codeword estimate:	$\hat{\mathbf{x}} = (\hat{x}_1, \ldots, \hat{x}_n) \in \hat{\mathcal{X}}^n$
Information word estimate:	$\hat{\mathbf{u}} = (\hat{u}_1, \ldots, \hat{u}_k) \in \hat{\mathcal{U}}^k$

Note:

- All the above alphabets **might be distinct**!
- Often, $\hat{\mathcal{X}} = \mathcal{X}$ and $\hat{\mathcal{U}} = \mathcal{U}$, but $\hat{\mathcal{X}} = \mathcal{X} \cup \{?\}$, etc., are also possible.

Some simple encoders

Simple Encoder: Example 1

Let $\mathcal{U} = \mathcal{X} = \mathbb{F}_2 = \{0,1\}$. (Here, $\mathbb{F}_2$ denotes the finite field with two elements.*)

Let $n = 3$ and $k = 1$.

Define the encoding mapping

$$E : \mathbb{F}_2^k \to \mathbb{F}_2^n \quad \text{with} \quad E(\mathbf{u}) \triangleq \begin{cases} (0,0,0) & \text{if } \mathbf{u} = (0) \\ (1,1,1) & \text{if } \mathbf{u} = (1) \end{cases}$$

In tabular form, the encoding mapping is

$\mathbf{u}$	$\mapsto$	$\mathbf{x}$
(0)	$\mapsto$	$(0,0,0)$
(1)	$\mapsto$	$(1,1,1)$

---

* Sometimes also denoted $\mathrm{GF}(2)$ and called the Galois field with two elements.

Simple Encoder: Example 1

In tabular form, the encoding mapping is

$\mathbf{u}$	$\mapsto$	$\mathbf{x}$
(0)	$\mapsto$	$(0,0,0)$
(1)	$\mapsto$	$(1,1,1)$

Defining the 1×3 matrix

$$\mathbf{G} \triangleq \begin{pmatrix} 1 & 1 & 1 \end{pmatrix},$$

one can verify that the encoding mapping can also be written as follows:

$$\mathbf{u} \mapsto \mathbf{x} \triangleq \mathbf{u} \cdot \mathbf{G},$$

i.e.,

$$\mathbb{C} = \{\mathbf{u} \cdot \mathbf{G} \mid \mathbf{u} \in \mathbb{F}_2^1\}.$$

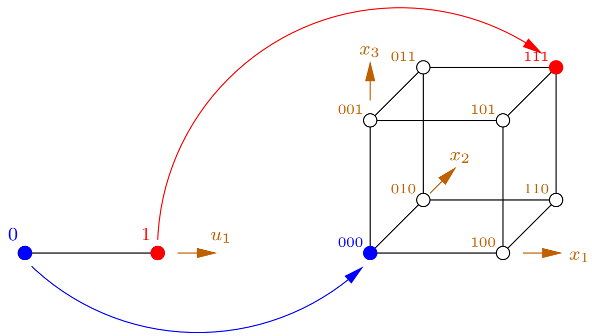
The matrix $\mathbf{G}$ is called a **generator matrix** for $\mathbb{C}$.

Simple Encoder: Example 1

In tabular form, the encoding mapping is

$\mathbf{u}$	$\mapsto$	$\mathbf{x}$
(0)	$\mapsto$	$(0,0,0)$
(1)	$\mapsto$	$(1,1,1)$

Graphically, the encoding mapping is



Simple Encoder: Example 1

In tabular form, the encoding mapping is

$\mathbf{u}$	$\mapsto$	$\mathbf{x}$
(0)	$\mapsto$	$(0,0,0)$
(1)	$\mapsto$	$(1,1,1)$

Defining the 2×3 matrix

$$\mathbf{H} \triangleq \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix},$$

one can verify that the code $\mathbb{C}$ can also be described as follows:

$$\mathbb{C} = \{\mathbf{x} \in \mathbb{F}_2^3 \mid \mathbf{H} \cdot \mathbf{x}^\mathsf{T} = \mathbf{0}^\mathsf{T}\}.$$

The matrix $\mathbf{H}$ is called a **parity-check matrix** for $\mathbb{C}$.

Simple Encoder: Example 2

Let $\mathcal{U} = \mathcal{X} = \mathbb{F}_2 = \{0, 1\}$.
Let $n = 3$ and $k = 2$.

Define the encoding mapping

$$E : \mathbb{F}_2^k \to \mathbb{F}_2^n \quad \text{with} \quad E(\mathbf{u}) \triangleq \begin{cases} (0,0,0) & \text{if } \mathbf{u} = (0,0) \\ (0,1,1) & \text{if } \mathbf{u} = (0,1) \\ (1,0,1) & \text{if } \mathbf{u} = (1,0) \\ (1,1,0) & \text{if } \mathbf{u} = (1,1) \end{cases}$$

In tabular form, the encoding mapping is

$\mathbf{u}$	$\mapsto$	$\mathbf{x}$
$(0,0)$	$\mapsto$	$(0,0,0)$
$(0,1)$	$\mapsto$	$(0,1,1)$
$(1,0)$	$\mapsto$	$(1,0,1)$
$(1,1)$	$\mapsto$	$(1,1,0)$

Simple Encoder: Example 2

In tabular form, the encoding mapping is

$\mathbf{u}$	$\mapsto$	$\mathbf{x}$
$(0,0)$	$\mapsto$	$(0,0,0)$
$(0,1)$	$\mapsto$	$(0,1,1)$
$(1,0)$	$\mapsto$	$(1,0,1)$
$(1,1)$	$\mapsto$	$(1,1,0)$

Defining the 2×3 matrix

$$\mathbf{G} \triangleq \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix},$$

one can verify that the encoding mapping can also be written as follows:

$$\mathbf{u} \quad \mapsto \quad \mathbf{x} \triangleq \mathbf{u} \cdot \mathbf{G},$$

i.e.,

$$\mathbb{C} = \{\mathbf{u} \cdot \mathbf{G} \mid \mathbf{u} \in \mathbb{F}_2^2\}.$$

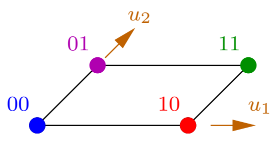
The matrix $\mathbf{G}$ is called a **generator matrix** for $\mathbb{C}$.

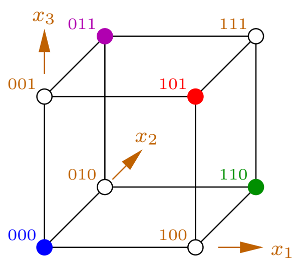
Simple Encoder: Example 2

In tabular form, the encoding mapping is

$\mathbf{u}$	$\mapsto$	$\mathbf{x}$
$(0,0)$	$\mapsto$	$(0,0,0)$
$(0,1)$	$\mapsto$	$(0,1,1)$
$(1,0)$	$\mapsto$	$(1,0,1)$
$(1,1)$	$\mapsto$	$(1,1,0)$

Graphically, the encoding mapping is





Simple Encoder: Example 2

In tabular form, the encoding mapping is

$\mathbf{u}$	$\mapsto$	$\mathbf{x}$
$(0,0)$	$\mapsto$	$(0,0,0)$
$(0,1)$	$\mapsto$	$(0,1,1)$
$(1,0)$	$\mapsto$	$(1,0,1)$
$(1,1)$	$\mapsto$	$(1,1,0)$

Defining the 1×3 matrix

$$\mathbf{H} \triangleq \begin{pmatrix} 1 & 1 & 1 \end{pmatrix},$$

one can verify that the code $\mathbb{C}$ can also be described as follows:

$$\mathbb{C} = \{\mathbf{x} \in \mathbb{F}_2^3 \mid \mathbf{H} \cdot \mathbf{x}^\mathsf{T} = \mathbf{0}^\mathsf{T}\}.$$

The matrix $\mathbf{H}$ is called a **parity-check matrix** for $\mathbb{C}$.

Comments w.r.t. Examples 1 and 2

- The codes in Examples 1 and 2 are called **linear codes** because the codes form subspaces of $\mathbb{F}_q^n$, i.e., any linear combination of codewords is again a codeword.
- The fact that the codes in Examples 1 and 2 are **linear codes** easily follows from their **description via a generator matrix** or their **description via a parity-check matrix**.
- In Examples 1 and 2, the mapping E is a **(strict-sense) systematic encoding mapping**, i.e.,

$$(x_1, \ldots x_k) = (u_1, \ldots, u_k) \quad \text{for all } \mathbf{x}, \mathbf{u} \text{ pairs.}$$

- In Examples 1 and 2, the codewords are **as far apart as possible** (under Hamming distance) for given code sizes.

Channel Models

In these lectures we will consider two main classes of channel models:

- Probabilistic channel models
- Adverserial channel models

Channel models

Probabilistic Channel Model

A probabilistic channel model is described by the conditional PMF / PDF

$$P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}).$$

In these lectures, we will often assume a **memoryless** channel (without feedback). With this,

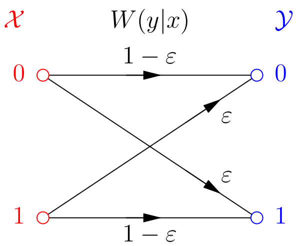
$$\begin{aligned} P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}) &= \prod_{i=1}^{n} P_{Y|X}(y_i|x_i) \\ &= \prod_{i=1}^{n} W(y_i|x_i), \end{aligned}$$

where we have introduced the **channel law** $W(y|x) \triangleq P_{Y|X}(y|x)$.

In the following slides, we will discuss some popular channel models.

The Binary Symmetric Channel

Let $\varepsilon \in [0, 1]$.



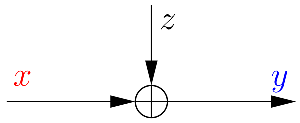
The $\mathrm{BSC}(\varepsilon)$, i.e., the binary symmetric channel with cross-over probability ε, is a discrete memoryless channel with

- input alphabet $\mathcal{X} = \{0,\ 1\}$,
- output alphabet $\mathcal{Y} = \{0,\ 1\}$,
- and conditional PMF

$$W(y|x) = \begin{cases} 1 - \varepsilon & (y = x) \\ \varepsilon & (y \neq x) \end{cases}.$$

Additive White Gaussian Noise Channel

Let $\sigma^2 \geq 0$.



The $\mathrm{AWGNC}(\sigma^2)$, the additive white Gaussian noise channel with noise variance σ^2, is a continuous-input continuous-output memoryless channel with

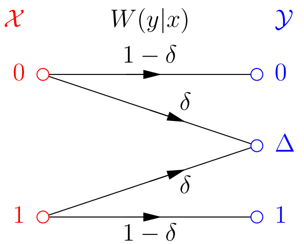
- input alphabet $\mathcal{X} = \mathbb{R}$,
- output alphabet $\mathcal{Y} = \mathbb{R}$,
- and conditional PDF

$$W(y|x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y - x)^2}{2\sigma^2}\right).$$

The channel output random variable is also given by $Y = X + Z$, where $Z \sim \mathcal{N}(0, \sigma^2)$ and where Z is statistically independent of X.

The Binary Erasure Channel

Let $\delta \in [0, 1]$.



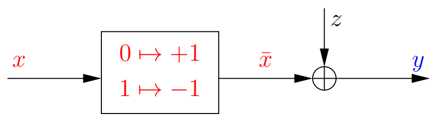
The $\mathrm{BEC}(\delta)$, the binary erasure channel with erasure probability δ, is a discrete memoryless channel with

- input alphabet $\mathcal{X} = \{0,\ 1\}$,
- output alphabet $\mathcal{Y} = \{0,\ \Delta,\ 1\}$,
- and conditional PMF

$$W(y|x) = \begin{cases} 1 - \delta & (y = x) \\ \delta & (y = \Delta) \end{cases}.$$

The Binary-Input Additive White Gaussian Noise Channel

Let $\sigma^2 \geq 0$.



The $\mathrm{BIAWGNC}(\sigma^2)$, the binary-input additive white Gaussian noise channel with noise variance σ^2, is a discrete-input continuous-output memoryless channel with

- input alphabet $\mathcal{X} = \{0, 1\}$,
- output alphabet $\mathcal{Y} = \mathbb{R}$,
- and conditional PDF

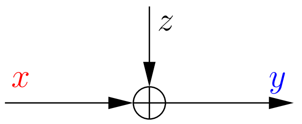
$$W(y|x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y - \bar{x})^2}{2\sigma^2}\right),$$

where

$$\bar{x} \triangleq 1 - 2x \triangleq \begin{cases} +1 & (x = 0) \\ -1 & (x = 1) \end{cases}.$$

The Binary Symmetric Channel (revisited)

Let $\varepsilon \in [0,1]$.



The $\mathrm{BSC}(\varepsilon)$ can also be defined as follows:

$$Y = X + Z,$$

where

- $P_Z(0) = 1 - \varepsilon$ and $P_Z(1) = \epsilon$,
- Z is statistically independent of X,
- X and Z are considered to be elements of $\mathbb{F}_2$ and so **addition is modulo** 2.

Note:

- For this channel model it makes sense to compare $\mathbf{x}$ and $\mathbf{y}$, to subtract $\mathbf{x}$ from $\mathbf{y}$, etc.
- In these lectures, we will often use the letter e instead of z.

Error detection and error correction

Adverserial Channel Models

Adverserial channel models are channel models where, for some given channel input vector, an adversary chooses

the "worst possible" channel output vector

among some channel-input-vector-dependent set.

Such channels are popular for cryptographic setups.

Error Detection and Error Correction

In the case of a BSC, we can write

$$\mathbf{y} = \mathbf{x} + \mathbf{e},$$

where the vector $\mathbf{e} = (e_1, \ldots, e_n)$ has components

$$e_i = y_i - x_i, \quad i = 1, \ldots, n.$$

Note:

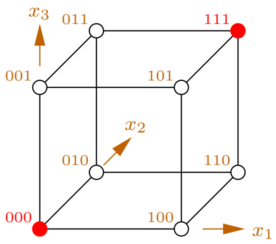
- Because $y_i, x_i \in \mathbb{F}_2$, the above **additions/subtractions are modulo** 2.
- Because $y_i, x_i \in \mathbb{F}_2$, we can also write $e_i = y_i - x_i$ as $e_i = y_i + x_i$.

Error detection: detect if $\mathbf{e} \neq \mathbf{0}$.
Error correction: we want to know $\mathbf{x}$ also if $\mathbf{e} \neq \mathbf{0}$.

Error Detecting Decoder

Consider the following setup:

- $\mathcal{U} = \mathcal{X} = \mathcal{Y} = \mathbb{F}_2$.
- $\hat{\mathcal{X}} = \{0, 1, \mathrm{err}\}$.
- The channel is a $\mathrm{BSC}(\varepsilon)$, $0 \le \varepsilon \le 1/2$.
- $\mathbb{C} = \{(0,0,0),\ (1,1,1)\}$.



An **error detecting decoder** is then given by

$$D_{\mathrm{DET}}(\mathbf{y}) \triangleq \begin{cases} (0,0,0) & \text{if } \mathbf{y} = (0,0,0) \\ (1,1,1) & \text{if } \mathbf{y} = (1,1,1) \\ (\mathrm{err},\mathrm{err},\mathrm{err}) & \text{otherwise} \end{cases}$$

Note: If $\mathbf{x} = (0,0,0)$ and $\mathbf{e} = (1,1,1)$, then $\mathbf{y} = (1,1,1)$ and $\hat{\mathbf{x}} \triangleq D_{\mathrm{DET}}(\mathbf{y}) = (1,1,1)$.

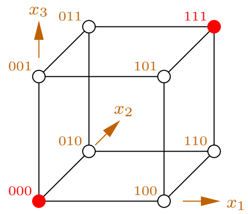
$\Rightarrow$ We do **not** detect that there were some errors!

$\Rightarrow$ In order to avoid this scenario as far as possible, codewords should be chosen to be **"as far apart as possible."**

Error Correcting Decoder

Consider the following setup:

- $\mathcal{U} = \mathcal{X} = \mathcal{Y} = \mathbb{F}_2$.
- $\hat{\mathcal{X}} = \{0, 1, ?\}$.
- The channel is a $\mathrm{BSC}(\varepsilon)$, $0 \le \varepsilon \le 1/2$.
- $\mathbb{C} = \{(0,0,0),\ (1,1,1)\}$.



An **error correcting decoder** is then given by

$$D_{\mathrm{DEC}}(\mathbf{y}) \triangleq \begin{cases} (0,0,0) & \text{if } \mathbf{y} \in \{(0,0,0),(0,0,1),(0,1,0),(1,0,0)\} \\ (1,1,1) & \text{if } \mathbf{y} \in \{(1,1,1),(1,1,0),(1,0,1),(0,1,1)\} \end{cases}$$

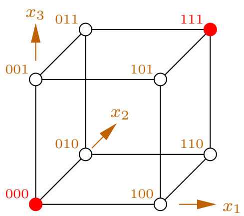
Note: If two of more symbol errors happen, the above decoder will fail.

$\Rightarrow$ In order to avoid this scenario as far as possible, codewords should be chosen to be **"as far apart as possible."**

Error Correcting Decoder

Consider the following setup:

- $\mathcal{U} = \mathcal{X} = \mathcal{Y} = \mathbb{F}_2$.
- $\hat{\mathcal{X}} = \{0, 1, ?\}$.
- The channel is a $\mathrm{BSC}(\varepsilon)$, $0 \le \varepsilon \le 1/2$.
- $\mathbb{C} = \{(0,0,0),\ (1,1,1)\}$.



An **error correcting decoder** is then given by

$$D_{\mathrm{DEC}}(\mathbf{y}) \triangleq \begin{cases} (0,0,0) & \text{if } \mathbf{y} \in \{(0,0,0),(0,0,1),(0,1,0),(1,0,0)\} \\ (1,1,1) & \text{if } \mathbf{y} \in \{(1,1,1),(1,1,0),(1,0,1),(0,1,1)\} \end{cases}$$

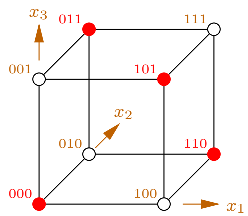
Note: The above decoder makes a **majority vote**, i.e.,

- if there are more 0s than 1s in $\mathbf{y}$, then $\hat{\mathbf{x}} = (0,0,0)$;
- if there are more 1s than 0s in $\mathbf{y}$, then $\hat{\mathbf{x}} = (1,1,1)$.

Error Correcting Decoder

Consider the following setup:

- $\mathcal{U} = \mathcal{X} = \mathcal{Y} = \mathbb{F}_2$.
- $\hat{\mathcal{X}} = \{0, 1, ?\}$.
- The channel is a $\mathrm{BSC}(\varepsilon)$, $0 \le \varepsilon \le 1/2$.
- $\mathbb{C} = \{(0,0,0),\ (0,1,1),\ (1,0,1),\ (1,1,0)\}$.



Note:

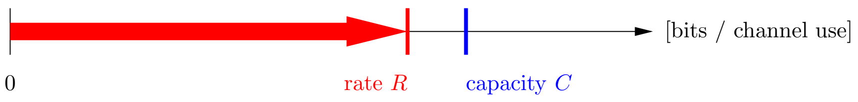
- If $\mathbf{x} = (0,0,0)$ and $\mathbf{e} = (1,0,0)$ then $\mathbf{y} = (1,0,0)$.
- If $\mathbf{x} = (1,0,1)$ and $\mathbf{e} = (0,0,1)$ then $\mathbf{y} = (1,0,0)$.
- If $\mathbf{x} = (1,1,0)$ and $\mathbf{e} = (0,1,0)$ then $\mathbf{y} = (1,0,0)$.

$\Rightarrow$ This code is **not strong enough** to **correct** a single symbol error.

$\Rightarrow$ However, it can **detect** a single symbol error.

What does information theory promise about channel coding?

Information Theory



- A channel is characterized by a number C called the capacity.
- A code is characterized by a number R called the rate.
- If $R < C$: there are codes, encoders, and decoders such that arbitrarily low error probabilities can be guaranteed (as long as one allows arbitrarily long codes).
- Shannon's proof was though non-constructive, i.e. it was not clear at all how to obtain specific well-performing finite-length codes that possess efficient encoders and decoders.

Information Theory

Shannon (1948): it is a good idea to use channel codes!

Some notation

Some Notation

Let $\mathcal{S}$ be a discrete or continuous set. Let $f : \mathcal{S} \to \mathbb{R}$ be some function.

Notation:

- The **maximum value of** f will be denoted by

$$\max_{s \in \mathcal{S}} f(s).$$

- The **set of locations where** f **takes on the maximum value** is

$$\left\{ s \in \mathcal{S} \;\middle|\; f(s) = \max_{s' \in \mathcal{S}} f(s') \right\},$$

  and will be denoted by

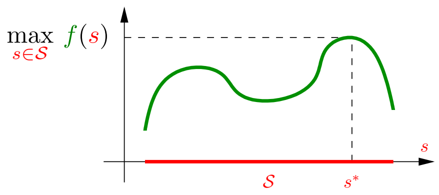
$$\arg\max_{s \in \mathcal{S}} f(s).$$

Some Notation

Note:

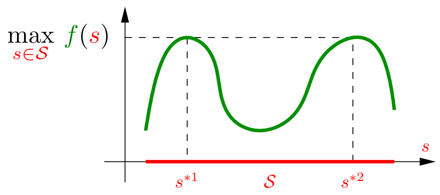
- The expression $\arg\max_{s \in \mathcal{S}} f(s)$ gives back a **set**!
- We will often assume that this set contains **only one element**, say s^*, and sloppily write expressions like

$$s^* \triangleq \arg\max_{s \in \mathcal{S}} f(s).$$



$$\arg\max_{s \in \mathcal{S}} f(s) = \{s^*\}$$

or $\arg\max_{s \in \mathcal{S}} f(s) = s^*$



$$\arg\max_{s \in \mathcal{S}} f(s) = \{s^{*1}, s^{*2}\}$$

Hamming distance and weight

Hamming Distance and Weight

Let $\mathcal{A}$ be some set and n a positive integer.

Definitions:

- The **Hamming distance** between two vectors $\mathbf{x}, \mathbf{y} \in \mathcal{A}^n$ is defined to be

$$d(\mathbf{x}, \mathbf{y}) \triangleq d_{\mathrm{H}}(\mathbf{x}, \mathbf{y}) \triangleq \big|\{i \mid x_i \neq y_i\}\big|.$$

- The **Hamming weight** of a vector $\mathbf{x} \in \mathcal{A}^n$ is defined to be

$$w(\mathbf{x}) \triangleq w_{\mathrm{H}}(\mathbf{x}) \triangleq \big|\{i \mid x_i \neq 0\}\big|.$$

  (Assumption: $0 \in \mathcal{A}$.)

Hamming Distance and Weight

Example 1: Let $\mathcal{A} \triangleq \{0,1\}$, $\mathbf{x} \triangleq (0,1,1)$, $\mathbf{y} \triangleq (1,0,1)$.

$\Rightarrow$ $d_{\mathrm{H}}(\mathbf{x},\mathbf{y}) = 2$.

$\Rightarrow$ $w_{\mathrm{H}}(\mathbf{x}) = 2$.

$\Rightarrow$ $w_{\mathrm{H}}(\mathbf{y}) = 2$.

Example 2: Let $\mathcal{A} \triangleq \{0,1,2\}$, $\mathbf{x} \triangleq (2,1,2,1,0)$, $\mathbf{y} \triangleq (1,2,2,0,0)$.

$\Rightarrow$ $d_{\mathrm{H}}(\mathbf{x},\mathbf{y}) = 3$.

$\Rightarrow$ $w_{\mathrm{H}}(\mathbf{x}) = 4$.

$\Rightarrow$ $w_{\mathrm{H}}(\mathbf{y}) = 3$.

Outline of Part 3

- Definition of blockwise ML decoding
- Blockwise ML decoding for BSC
- Blockwise ML decoding as solving an integer linear program
- Blockwise ML decoding as solving a linear program

Part 3

Maximum-likelihood (ML) decoding

Definition of blockwise ML decoding

Blockwise ML Decoding

Assumptions:

- The channel is described by $P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x})$.
- The code $\mathbb{C}$ is used.

Definition: Blockwise maximum-likelihood (ML) decoding of the received vector $\mathbf{y}$ yields the codeword estimate

$$\hat{\mathbf{x}}_{\mathrm{ML}}(\mathbf{y}) \triangleq \arg\max_{\mathbf{x}\in\mathbb{C}} P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}).$$

Note: If all codewords are sent equally likely, then

$\hat{\mathbf{x}}_{\mathrm{ML}}$ minimizes the block error probability,

i.e.,

$\hat{\mathbf{x}}_{\mathrm{ML}}$ minimizes $\Pr(\hat{\mathbf{x}}_{\mathrm{ML}}(\mathbf{Y}) \neq \mathbf{X})$.

Proof: Omitted.

Blockwise ML Decoding for BSC

Definition (reminder): Blockwise maximum-likelihood (ML) decoding of the received vector $\mathbf{y}$ yields the codeword estimate

$$\hat{\mathbf{x}}_{\mathrm{ML}}(\mathbf{y}) \triangleq \arg\max_{\mathbf{x}\in\mathbb{C}} P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}).$$

Theorem: Assume that the channel is a $\mathrm{BSC}(\varepsilon)$, with $0 \leq \varepsilon < 1/2$. Then

$$\hat{\mathbf{x}}_{\mathrm{ML}}(\mathbf{y}) = \arg\min_{\mathbf{x}\in\mathbb{C}} d_{\mathrm{H}}(\mathbf{x},\mathbf{y}).$$

Note: Interestingly enough, the right-hand side of the above expression is **independent of** ε as long as $0 \leq \varepsilon < 1/2$.

Blockwise ML Decoding

Note: Besides blockwise ML decoding, there are also

- symbolwise ML decoding,
- blockwise MAP decoding,
- symbolwise MAP decoding.

They all have their uses and are optimal in some suitable sense, but we will not talk more about them in these lectures.

(MAP: maximum a-posteriori)

Blockwise ML Decoding for BSC

Proof:

$$\begin{aligned}
\hat{\mathbf{x}}_{\mathrm{ML}}(\mathbf{y}) &\triangleq \arg\max_{\mathbf{x}\in\mathbb{C}} P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}) \\
&= \arg\max_{\mathbf{x}\in\mathbb{C}} \prod_{i=1}^{n} W(y_i|x_i) \\
&\overset{\text{(a)}}{=} \arg\max_{\mathbf{x}\in\mathbb{C}} \log\left(\prod_{i=1}^{n} W(y_i|x_i)\right) \\
&= \arg\max_{\mathbf{x}\in\mathbb{C}} \sum_{i=1}^{n} \log\left(W(y_i|x_i)\right) \\
&\overset{\text{(b)}}{=} \arg\max_{\mathbf{x}\in\mathbb{C}} \left(n - d_{\mathrm{H}}(\mathbf{x},\mathbf{y})\right)\cdot\log(1-\varepsilon) + d_{\mathrm{H}}(\mathbf{x},\mathbf{y})\cdot\log(\varepsilon) \\
&= \arg\max_{\mathbf{x}\in\mathbb{C}} n\cdot\log(1-\varepsilon) - d_{\mathrm{H}}(\mathbf{x},\mathbf{y})\cdot\log\left(\frac{1-\varepsilon}{\varepsilon}\right) \\
&= \arg\max_{\mathbf{x}\in\mathbb{C}} -d_{\mathrm{H}}(\mathbf{x},\mathbf{y})\cdot\log\left(\frac{1-\varepsilon}{\varepsilon}\right) \\
&\overset{\text{(c)}}{=} \arg\min_{\mathbf{x}\in\mathbb{C}} d_{\mathrm{H}}(\mathbf{x},\mathbf{y}).
\end{aligned}$$

Blockwise ML Decoding for BSC

Proof (continued):

- Step (a) follows from the fact that $\log(\,\cdot\,)$ is a strictly increasing function.
- Step (b) follows from

$$\log\left(W(y_i|x_i)\right) = \begin{cases} \log(1-\varepsilon) & \text{if } y_i = x_i \\ \log(\varepsilon) & \text{if } y_i \neq x_i \end{cases}$$

- Step (c) follows from

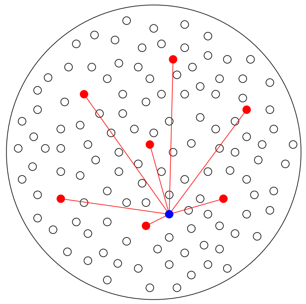
$$\frac{1-\varepsilon}{\varepsilon} > 1\,,$$

which implies

$$\log\left(\frac{1-\varepsilon}{\varepsilon}\right) > 0\,.$$

Blockwise ML Decoding for BSC

Geometric picture for $\hat{\mathbf{x}}_{\mathrm{ML}}(\mathbf{y}) \triangleq \arg\min_{\mathbf{x}\in\mathbb{C}} d_{\mathrm{H}}(\mathbf{x},\mathbf{y})$:



- ∘ points in $\mathcal{X}^n$
- • codewords, i.e., points in $\mathbb{C}$
- • received vector $\mathbf{y}$

The expression $\arg\min_{\mathbf{x}\in\mathbb{C}} d_{\mathrm{H}}(\mathbf{x},\mathbf{y})$ means the following:

- Compute $d_{\mathrm{H}}(\mathbf{x},\mathbf{y})$ for every $\mathbf{x}\in\mathbb{C}$.
- Take the $\mathbf{x}\in\mathbb{C}$ for which $d_{\mathrm{H}}(\mathbf{x},\mathbf{y})$ is minimized.
- If there is a **tie**, we can either declare failure or randomly pick one of the optimal codewords.

Blockwise ML Decoding for BSC

Note: Many papers on coding theory **start** with the

minimum-distance decoding rule $\quad \hat{\mathbf{x}}(\mathbf{y}) \triangleq \arg\min_{\mathbf{x}\in\mathbb{C}} d_{\mathrm{H}}(\mathbf{x},\mathbf{y})$.

However, the **minimum-distance decoding rule** is optimal only for certain setups, like the setup in the above theorem. In general, it is only a **decoding heuristic** (often a good one) for channels with $\mathcal{Y} = \mathcal{X}$.

The minimum-distance decoding rule can even be "totally useless"!
For example, for a $\mathrm{BSC}(\varepsilon)$ with $1/2 < \varepsilon \leq 1$ one obtains

$$\hat{\mathbf{x}}_{\mathrm{ML}}(\mathbf{y}) \triangleq \arg\max_{\mathbf{x}\in\mathbb{C}} P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}) = \arg\max_{\mathbf{x}\in\mathbb{C}} d_{\mathrm{H}}(\mathbf{x},\mathbf{y}).$$

(This is a consequence of $\log\left(\frac{1-\varepsilon}{\varepsilon}\right) < 0$.)

Blockwise ML Decoding for BSC

Example: minimum-distance decoding for

$$\mathbb{C} \triangleq \{(0,0,0,0,0),\ (1,1,1,0,0),\ (0,0,1,1,1),\ (1,1,0,1,1)\}.$$

Assume that the transmitted codeword is $\mathbf{x} = (0,0,1,1,1)$.

	Scenario 1	**Scenario 2**	**Scenario 3**
	$\mathbf{e} = (0,1,0,0,0)$ $\to \mathbf{y} = (0,1,1,1,1)$	$\mathbf{e} = (0,1,0,0,1)$ $\to \mathbf{y} = (0,1,1,1,0)$	$\mathbf{e} = (1,1,0,0,0)$ $\to \mathbf{y} = (1,1,1,1,1)$
$d_{\mathrm{H}}((0,0,0,0,0),\mathbf{y})$	$= 4$	$= 3$	$= 5$
$d_{\mathrm{H}}((1,1,1,0,0),\mathbf{y})$	$= 3$	$= 2$	$= 2$
$d_{\mathrm{H}}((0,0,1,1,1),\mathbf{y})$	$= 1$	$= 2$	$= 2$
$d_{\mathrm{H}}((1,1,0,1,1),\mathbf{y})$	$= 2$	$= 3$	$= 1$
Comment	$\hat{\mathbf{x}} = \mathbf{x}$	tie!	$\hat{\mathbf{x}} \neq \mathbf{x}$

As we will see later on, this code has $d_{\min}(\mathbb{C}) = 3$ and so **one bit flip** will be correctly decoded by a minimum-distance decoder.
Potentially, a minimum-distance decoder can correct more bit flips, but there is no guarantee.

ML decoding as solving a linear program

ML Decoding as an *Integer* LP

Derivation (we assume to have a memoryless channel):

$$
\begin{aligned}
\arg\max_{\mathbf{x}\in\mathbb{C}} P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x})
&= \arg\max_{\mathbf{x}\in\mathbb{C}} \log \prod_{i=1}^{n} P_{Y_i|X_i}(y_i|x_i) \\
&= \arg\max_{\mathbf{x}\in\mathbb{C}} \sum_{i=1}^{n} \log P_{Y_i|X_i}(y_i|x_i) \\
&= \arg\max_{\mathbf{x}\in\mathbb{C}} \sum_{i=1}^{n} \left(x_i \log \frac{P_{Y_i|X_i}(y_i|1)}{P_{Y_i|X_i}(y_i|0)} + \log P_{Y_i|X_i}(y_i|0) \right) \\
&= \arg\max_{\mathbf{x}\in\mathbb{C}} \sum_{i=1}^{n} x_i(-\lambda_i) = \arg\min_{\mathbf{x}\in\mathbb{C}} \sum_{i=1}^{n} x_i\lambda_i.
\end{aligned}
$$

ML Decoding as an *Integer* LP

For memoryless channels, blockwise ML decoding of a binary code can be written as an integer linear program.

$$\hat{\mathbf{x}}_{\mathrm{ML}}^{\mathrm{block}}(\mathbf{y}) = \arg\max_{\mathbf{x}\in\mathbb{C}} P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}) = \arg\min_{\mathbf{x}\in\mathbb{C}} \sum_{i=1}^{n} x_i\lambda_i,$$

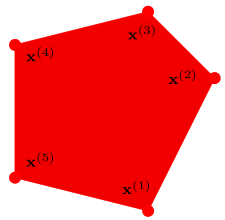
where

$$\lambda_i \triangleq \lambda_i(y_i) \triangleq \log \frac{P_{Y|X}(y_i|0)}{P_{Y|X}(y_i|1)}.$$

ML Decoding as an LP

$$\arg\min_{\mathbf{x}\in\mathbb{C}} \sum_{i=1}^{n} \lambda_i x_i$$

$$\overset{*}{=} \arg \min_{\mathbf{x}\in\mathrm{conv}(\mathbb{C})} \sum_{i=1}^{n} \lambda_i x_i$$



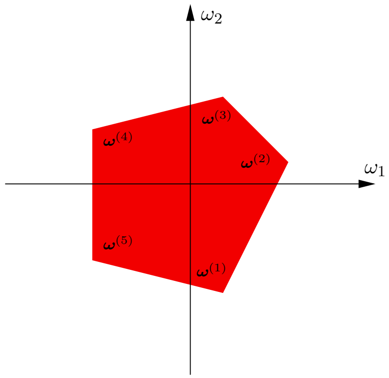
$\overset{*}{=}$ sign: This is an equality if there is a unique $\mathbf{x} \in \mathbb{C}$ that minimizes $\sum_{i=1}^{n} \lambda_i x_i$. Otherwise, the left-hand side is a subset of the right-hand side.

e.g.,

$$\mathbb{C} = \left\{ \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(5)} \right\}$$

Linear Programs (LPs)

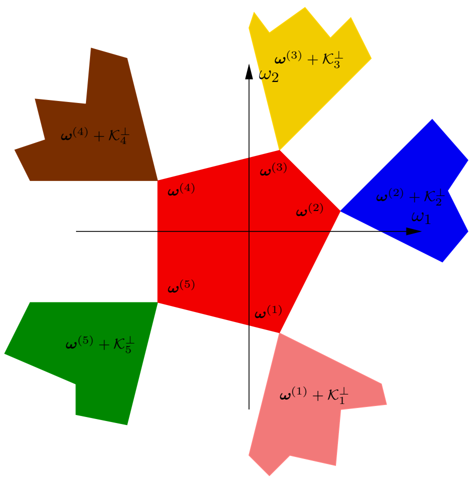
$$\arg\max_{\omega \in \mathcal{A}} \sum_{i=1}^{n} c_i \omega_i$$

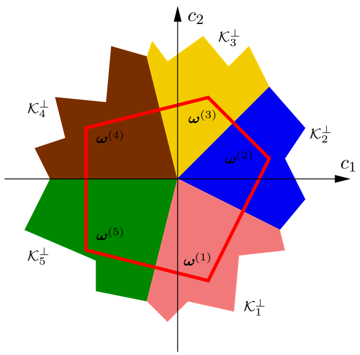


Part 4

Linear-programming decoding

Linear Programs (LPs)





$$\arg\max_{\omega \in \mathcal{A}} \sum_{i=1}^{n} c_i \omega_i$$

Outline of Part 4

- From blockwise ML decoding to linear-programming decoding
- The fundamental polytope and the fundamental cone
- ML Certificate Property

From blockwise ML decoding

to linear-programming decoding

ML Decoding as an LP

$$\hat{\mathbf{x}}_{\mathrm{ML}}^{\mathrm{block}}(\mathbf{y}) = \arg \min_{\mathbf{x} \in \mathrm{conv}(\mathbb{C})} \sum_{i=1}^{n} x_i \lambda_i ,$$

This is a linear program.

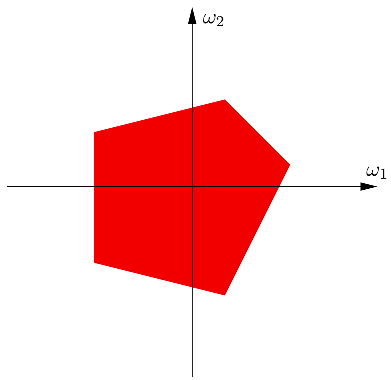
However, the

number of variables / equalities / inequalities

needed to describe the polytope $\mathrm{conv}(\mathbb{C})$ is (usually) exponential in n.

Relaxed Linear Programs

$$\arg \max_{\omega \in \mathcal{A}} \sum_{i=1}^{n} c_i \omega_i$$

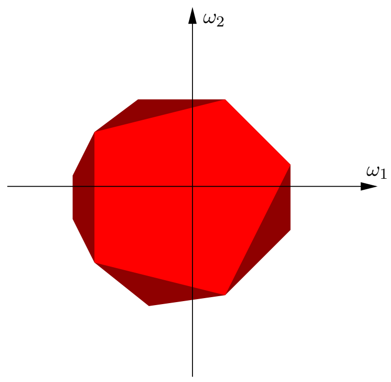


Relaxed Linear Programs

$$\arg \max_{\omega \in \mathcal{A}} \sum_{i=1}^{n} c_i \omega_i$$

is replaced by

$$\arg \max_{\omega \in \mathcal{A}'} \sum_{i=1}^{n} c_i \omega_i$$

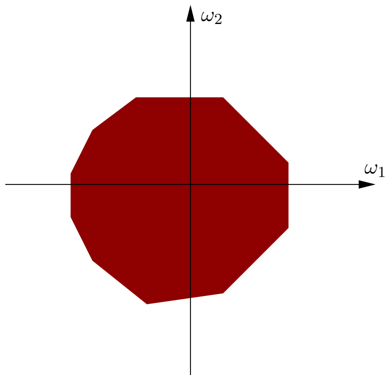


Relaxed Linear Programs

$$\arg\max_{\omega \in \mathcal{A}} \sum_{i=1}^{n} c_i \omega_i$$

is replaced by

$$\arg\max_{\omega \in \mathcal{A}'} \sum_{i=1}^{n} c_i \omega_i$$

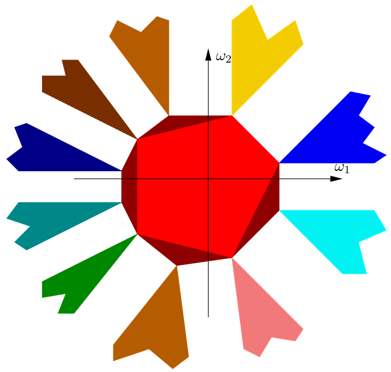


Relaxed Linear Programs

$$\arg\max_{\omega \in \mathcal{A}} \sum_{i=1}^{n} c_i \omega_i$$

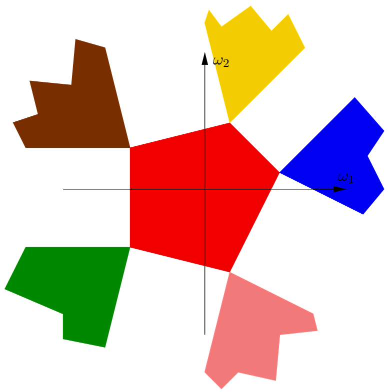
is replaced by

$$\arg\max_{\omega \in \mathcal{A}'} \sum_{i=1}^{n} c_i \omega_i$$



Relaxed Linear Programs

$$\arg\max_{\omega \in \mathcal{A}} \sum_{i=1}^{n} c_i \omega_i$$



LP Decoding

$$\hat{\boldsymbol{\omega}}_{\mathrm{ML}}^{\mathrm{block}}(\mathbf{y}) = \arg \min_{\omega \in \mathrm{conv}(\mathbb{C})} \sum_{i=1}^{n} \omega_i \lambda_i .$$

A standard approach in optimization theory is then to relax the set $\mathrm{conv}(\mathbb{C})$ to a set $\mathrm{relax}(\mathrm{conv}(\mathbb{C}))$ whose description complexity is much lower:

$$\hat{\boldsymbol{\omega}}_{\mathrm{LP}}(\mathbf{y}) = \arg \min_{\omega \in \mathrm{relax}(\mathrm{conv}(\mathbb{C}))} \sum_{i=1}^{n} \omega_i \lambda_i .$$

Linear Programming Decoding

How do we obtain a suitable relaxation? The following approach was proposed by Feldman / Karger / Wainwright and seems to work well for low-density parity-check (LDPC) codes.

Before showing how this relaxation works, let us remember how we define a code using a parity-check matrix.
Let $\mathbf{H}$ be a parity-check matrix, e.g.,

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix}.$$

A vector $\mathbf{x} \in \mathbb{F}_2^5$ is a codeword if and only if

$$\mathbf{H}\mathbf{x}^\mathsf{T} = \mathbf{0}^\mathsf{T}.$$

Linear Programming Decoding

Let the relaxation $\mathrm{relax}(\mathrm{conv}(\mathbb{C}))$ of $\mathrm{conv}(\mathbb{C})$ be the set of all vectors $\boldsymbol{\omega} \in \mathbb{R}^5$ that fulfill three conditions:

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix} \quad \Rightarrow \quad \begin{array}{l} \boldsymbol{\omega} \in \mathrm{conv}(\mathbb{C}_1) \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathbb{C}_2) \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathbb{C}_3) \end{array}$$

Therefore,

$$\mathbb{C} \subset \mathrm{conv}(\mathbb{C}) \subseteq \mathrm{relax}(\mathrm{conv}(\mathbb{C})) \triangleq \underbrace{\mathrm{conv}(\mathbb{C}_1) \cap \mathrm{conv}(\mathbb{C}_2) \cap \mathrm{conv}(\mathbb{C}_3)}_{\text{Fundamental polytope } \mathcal{P}(\mathbf{H})}.$$

This relaxation turns out to have many desirable properties. Note that the points in $\mathcal{P}(\mathbf{H})$ are called pseudo-codewords.

Linear Programming Decoding

In our case this means that $\mathbf{x}$ is a codeword if and only if $\mathbf{x}$ fulfills the following three equations:

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix} \quad \Rightarrow \quad \begin{array}{l} x_1 + x_2 + x_3 = 0 \pmod 2 \\ x_2 + x_4 + x_5 = 0 \pmod 2 \\ x_3 + x_4 + x_5 = 0 \pmod 2 \end{array}$$

Therefore, $\mathbb{C}$ can be seen as the intersection of three codes

$$\mathbb{C} = \mathbb{C}_1 \cap \mathbb{C}_2 \cap \mathbb{C}_3,$$

where

$$\mathbb{C}_1 \triangleq \left\{ \mathbf{x} \in \mathbb{F}_2^5 \,\middle|\, \mathbf{h}_1 \mathbf{x}^\mathsf{T} = 0 \pmod 2 \right\},$$
$$\mathbb{C}_2 \triangleq \left\{ \mathbf{x} \in \mathbb{F}_2^5 \,\middle|\, \mathbf{h}_2 \mathbf{x}^\mathsf{T} = 0 \pmod 2 \right\},$$
$$\mathbb{C}_3 \triangleq \left\{ \mathbf{x} \in \mathbb{F}_2^5 \,\middle|\, \mathbf{h}_3 \mathbf{x}^\mathsf{T} = 0 \pmod 2 \right\}.$$

Blockwise ML Decoding vs. LP Decoding

Blockwise ML decoding:

$$\hat{\mathbf{x}}_{\mathrm{ML}}^{\mathrm{block}}(\mathbf{y}) = \arg \min_{\mathbf{x} \in \mathrm{conv}(\mathbb{C})} \sum_{i=1}^{n} x_i \lambda_i.$$

LP decoding:

$$\hat{\boldsymbol{\omega}}_{\mathrm{LP}}(\mathbf{y}) = \arg \min_{\boldsymbol{\omega} \in \mathcal{P}(\mathbf{H})} \sum_{i=1}^{n} \omega_i \lambda_i.$$

Blockwise ML Decoding vs. LP Decoding

Blockwise ML decoding:

$$\hat{\mathbf{x}}^{\text{block}}_{\text{ML}}(\mathbf{y}) = \arg \min_{\mathbf{x} \in \text{conv}\left(\cap_{j=1}^{m} \mathbb{C}_j\right)} \sum_{i=1}^{n} x_i \lambda_i .$$

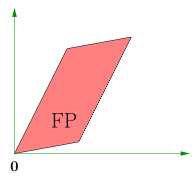
LP decoding:

$$\hat{\boldsymbol{\omega}}_{\text{LP}}(\mathbf{y}) = \arg \min_{\boldsymbol{\omega} \in \cap_{j=1}^{m} \text{conv}(\mathbb{C}_j)} \sum_{i=1}^{n} \omega_i \lambda_i .$$

Fundamental Polytope

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix} \begin{matrix} \Rightarrow \mathbb{C}_1 & \quad \Rightarrow \text{conv}(\mathbb{C}_1) \\ \Rightarrow \mathbb{C}_2 & \quad \Rightarrow \text{conv}(\mathbb{C}_2) \\ \Rightarrow \mathbb{C}_3 & \quad \Rightarrow \text{conv}(\mathbb{C}_3) \end{matrix}$$

$$\Rightarrow \mathbb{C} = \bigcap_{j=1}^{m} \mathbb{C}_j \qquad \Rightarrow \underbrace{\mathcal{P}(\mathbf{H}) = \bigcap_{j=1}^{m} \text{conv}(\mathbb{C}_j)}_{\text{Fundamental polytope}}$$

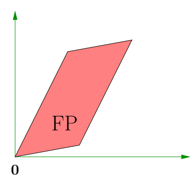


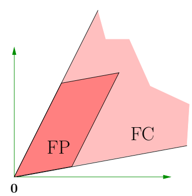
Fundamental polytope and fundamental code

Fundamental Polytope / Cone

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix} \begin{matrix} \Rightarrow \text{conv}(\mathbb{C}_1) & \quad \Rightarrow \text{conic}(\mathbb{C}_1) \\ \Rightarrow \text{conv}(\mathbb{C}_2) & \quad \Rightarrow \text{conic}(\mathbb{C}_2) \\ \Rightarrow \text{conv}(\mathbb{C}_3) & \quad \Rightarrow \text{conic}(\mathbb{C}_3) \end{matrix}$$

$$\Rightarrow \underbrace{\mathcal{P}(\mathbf{H}) = \bigcap_{j=1}^{m} \text{conv}(\mathbb{C}_j)}_{\text{Fundamental polytope}} \qquad \Rightarrow \underbrace{\mathcal{K}(\mathbf{H}) = \bigcap_{j=1}^{m} \text{conic}(\mathbb{C}_j)}_{\text{Fundamental cone}}$$





Fundamental Polytope / Cone

Note: because for binary-input output-symmetric channels the analysis of the fundamental polytope essentially boils down to the analysis of the fundamental cone, all the points in the fundamental cone will also be called pseudo-codewords.

Convex Hull of Simple Codes

Let $\mathbb{C}$ be defined by the parity-check matrix

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}.$$

Then

$$\mathbb{C} = \{(0,0,0),\ (1,1,0),\ (1,0,1),\ (0,1,1)\}$$

and

$$\mathrm{conv}(\mathbb{C}) = \left\{ \boldsymbol{\omega} \in [0,1]^3 \;\middle|\; \begin{array}{l} -\omega_1+\omega_2+\omega_3 \geq 0 \\ +\omega_1-\omega_2+\omega_3 \geq 0 \\ +\omega_1+\omega_2-\omega_3 \geq 0 \\ -\omega_1-\omega_2-\omega_3 \geq -2 \end{array} \right\}.$$

Convex Hull of Simple Codes

Let $\mathbb{C}$ be defined by the parity-check matrix

$$\mathbf{H} = \begin{pmatrix} 1 & 1 \end{pmatrix}.$$

Then

$$\mathbb{C} = \{(0,0),\ (1,1)\}$$

and

$$\mathrm{conv}(\mathbb{C}) = \left\{ \boldsymbol{\omega} \in [0,1]^2 \;\middle|\; \begin{array}{l} -\omega_1+\omega_2 \geq 0 \\ +\omega_1-\omega_2 \geq 0 \end{array} \right\},$$

where $[0,1] = \{r \in \mathbb{R} \,|\, 0 \leq r \leq 1\}$.

Conic Hull of Simple Codes

Let $\mathbb{C}$ be defined by the parity-check matrix

$$\mathbf{H} = \begin{pmatrix} 1 & 1 \end{pmatrix}.$$

Then

$$\mathbb{C} = \{(0,0),\ (1,1)\}$$

and

$$\mathrm{conic}(\mathbb{C}) = \left\{ \boldsymbol{\omega} \in \mathbb{R}_+^2 \;\middle|\; \begin{array}{l} -\omega_1+\omega_2 \geq 0 \\ +\omega_1-\omega_2 \geq 0 \end{array} \right\},$$

where $\mathbb{R}_+ = \{r \in \mathbb{R} \,|\, r \geq 0\}$.

Conic Hull of Simple Codes

Let $\mathbb{C}$ be defined by the parity-check matrix

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}.$$

Then

$$\mathbb{C} = \big\{(0,0,0),\ (1,1,0),\ (1,0,1),\ (0,1,1)\big\}$$

and

$$\operatorname{conic}(\mathbb{C}) = \left\{ \boldsymbol{\omega} \in \mathbb{R}_+^3 \;\middle|\; \begin{array}{l} -\omega_1+\omega_2+\omega_3 \geq 0 \\ +\omega_1-\omega_2+\omega_3 \geq 0 \\ +\omega_1+\omega_2-\omega_3 \geq 0 \end{array} \right\}.$$

A Simple Code

The fundamental polytope is $\mathcal{P}(\mathbf{H}) = \operatorname{conv}(\mathbb{C}_1) \cap \operatorname{conv}(\mathbb{C}_2) \cap \operatorname{conv}(\mathbb{C}_3)$ with

$$\begin{aligned}
\operatorname{conv}(\mathbb{C}_1) &= \operatorname{conv}\Big(\big\{(0,0,0),\ (1,1,0),\ (0,0,1),\ (1,1,1)\big\}\Big) \\
&= \left\{ \boldsymbol{\omega} \in [0,1]^3 \;\middle|\; \begin{array}{l} -\omega_1+\omega_2 \geq 0 \\ +\omega_1-\omega_2 \geq 0 \end{array} \right\} \\
\operatorname{conv}(\mathbb{C}_2) &= \operatorname{conv}\Big(\big\{(0,0,0),\ (1,1,0),\ (1,0,1),\ (0,1,1)\big\}\Big) \\
&= \left\{ \boldsymbol{\omega} \in [0,1]^3 \;\middle|\; \begin{array}{l} -\omega_1+\omega_2+\omega_3 \geq 0 \\ +\omega_1-\omega_2+\omega_3 \geq 0 \\ +\omega_1+\omega_2-\omega_3 \geq 0 \\ -\omega_1-\omega_2-\omega_3 \geq -2 \end{array} \right\} \\
\operatorname{conv}(\mathbb{C}_3) &= \operatorname{conv}\Big(\big\{(0,0,0),\ (0,1,1),\ (1,0,0),\ (1,1,1)\big\}\Big) \\
&= \left\{ \boldsymbol{\omega} \in [0,1]^3 \;\middle|\; \begin{array}{l} -\omega_2+\omega_3 \geq 0 \\ +\omega_2-\omega_3 \geq 0 \end{array} \right\}
\end{aligned}$$

A Simple Code

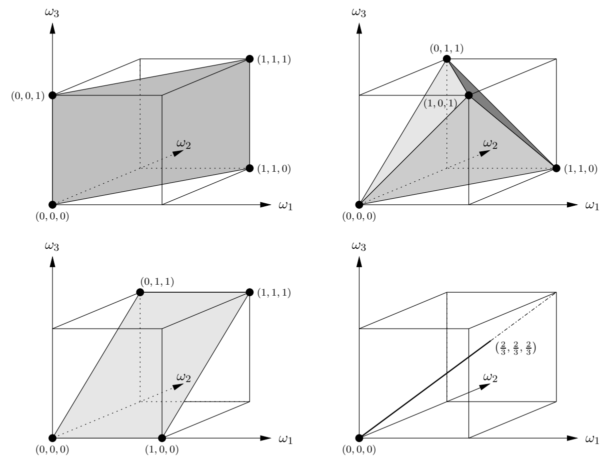
Let us consider the length-3 code $\mathbb{C}$ defined by the parity-check matrix

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}.$$

The code $\mathbb{C}$ can be written as $\mathbb{C} = \mathbb{C}_1 \cap \mathbb{C}_2 \cap \mathbb{C}_3$ with

$$\begin{aligned}
\mathbb{C}_1 &= \big\{(0,0,0),\ (1,1,0),\ (0,0,1),\ (1,1,1)\big\} \\
\mathbb{C}_2 &= \big\{(0,0,0),\ (1,1,0),\ (1,0,1),\ (0,1,1)\big\} \\
\mathbb{C}_3 &= \big\{(0,0,0),\ (0,1,1),\ (1,0,0),\ (1,1,1)\big\}
\end{aligned}$$

A Simple Code



$\operatorname{conv}(\mathbb{C}_1)$	$\operatorname{conv}(\mathbb{C}_2)$
$\operatorname{conv}(\mathbb{C}_3)$	$\mathcal{P}(\mathbf{H})$

ML certificate property

Equivalence of decoders for the BEC

ML Certificate Property

Theorem:

LP decoding has the ML certificate property:

if LP decoding outputs a codeword,
it is guaranteed to be the blockwise ML codeword.

Note: This does **not** mean that if LP decoding outputs a codeword that LP decoding was successful. The reason for this is that blockwise ML decoding might fail, i.e., output a codeword that is different from the transmitted codeword.

Equivalence of Decoders for the BEC

For the BEC, the following decoders give the same decoding result:

- sum-product algorithm (SPA) decoding,*
- max-product algorithm (MPA) decoding,*
- peeling decoding,*
- linear programming (LP) decoding,
- symbol-wise graph-cover decoding,
- block-wise graph-cover decoding.

Proof: Omitted.

* After convergence. For the BEC, one can show that SPA decoding (with flooding schedule), MPA decoding (with flooding schedule), and the peeling decoder converge in a finite number of iterations. (SPA decoding and MPA decoding converge after the same number of iterations, but the the peeling decoder might converge after a different number of iterations.)

References (1/2)

LP decoding was introduced by Feldman, Wainwright, and Karger:

- J. Feldman, *Decoding Error-Correcting Codes via Linear Programming*, Ph.D. thesis, Dept. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, 2003.
- J. Feldman, M. J. Wainwright and D. R. Karger, "Using linear programming to decode binary linear codes," IEEE Trans. Inf. Theory, vol. 51, no. 3, pp. 954–972, Mar. 2005.

The relaxed polytope introduced by Feldman, Wainwright, and Karger happened to be equivalent to the fundamental polytope introduced in a different context by Koetter and Vontobel, and nowadays the relaxed polytope in LP decoding is typically called the fundamental polytope.

- R. Koetter and P. O. Vontobel, "Graph covers and iterative decoding of finite-length codes," Proc. 3rd Intern. Symp. on Turbo Codes and Related Topics, Brest, France, pp. 75–82, Sep. 1–5, 2003.

Part 5

Graphical representation of codes

References (2/2)

The notion of LP decoding appears also in the context of compressed sensing:

- E. J. Candes and T. Tao, "Decoding by linear programming," IEEE Trans. Inf. Theory, vol. 51, no. 12, pp. 4203–4215, Dec. 2005.

This notion of LP decoding is rather different than the notion of LP decoding of LDPC codes as discussed in these slides, but there are mathematical connections between the two, as explained in the following paper:

- A. Dimakis, R. Smarandache, and P. O. Vontobel, "LDPC codes for compressed sensing," IEEE Trans. Inf. Theory, vol. 58, no. 5, pp. 3093–3114, May 2012.

In particular, this paper shows how to construct

"good" compressed sensing measurement matrices

based on

"good" low-density parity-check matrices.

Outline of Part 5

- Graphical representation of codes
- Another example for graphical representation of a code
- Graphical representation of codeword indicator function and pseudo-codeword indicator function

Binary Linear Codes

Let $\mathbf{H}$ be a parity-check matrix, e.g.,

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \end{pmatrix}.$$

The code $\mathbb{C}$ described by $\mathbf{H}$ is then

$$\mathbb{C} = \left\{ (x_1, x_2, x_3, x_4, x_5) \in \mathbb{F}_2^5 \,\middle|\, \mathbf{H} \cdot \mathbf{x}^\mathsf{T} = \mathbf{0}^\mathsf{T} \pmod 2 \right\}.$$

A vector $\mathbf{x} \in \mathbb{F}_2^5$ is a codeword if and only if

$$\mathbf{H} \cdot \mathbf{x}^\mathsf{T} = \mathbf{0}^\mathsf{T} \pmod 2.$$

Binary Linear Codes

Defining the codes $\mathbb{C}_1$ and $\mathbb{C}_2$ where

$$\mathbb{C}_1 = \left\{ (x_1, x_2, x_3, x_4, x_5) \in \mathbb{F}_2^5 \,\middle|\, x_1 + x_2 + x_3 = 0 \pmod 2 \right\},$$
$$\mathbb{C}_2 = \left\{ (x_1, x_2, x_3, x_4, x_5) \in \mathbb{F}_2^5 \,\middle|\, x_2 + x_4 + x_5 = 0 \pmod 2 \right\},$$

the code $\mathbb{C}$ can be written as the intersection of $\mathbb{C}_1$ and $\mathbb{C}_2$:

$$\mathbb{C} = \mathbb{C}_1 \cap \mathbb{C}_2.$$

Binary Linear Codes

This means that $\mathbf{x}$ is a codeword if and only if $\mathbf{x}$ fulfills the following two equations:

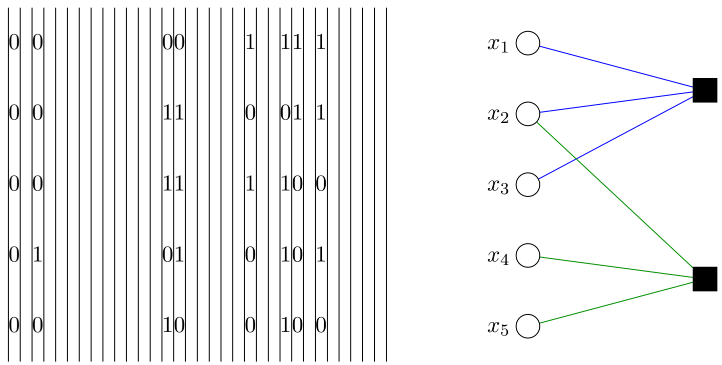
$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \end{pmatrix} \quad \Rightarrow \quad \begin{array}{l} x_1 + x_2 + x_3 = 0 \pmod 2 \\ x_2 + x_4 + x_5 = 0 \pmod 2 \end{array}$$

In summary,

$$\begin{aligned} \mathbb{C} &= \left\{ (x_1, x_2, x_3, x_4, x_5) \in \mathbb{F}_2^5 \,\middle|\, \mathbf{H} \cdot \mathbf{x}^\mathsf{T} = \mathbf{0}^\mathsf{T} \pmod 2 \right\} \\ &= \left\{ (x_1, x_2, x_3, x_4, x_5) \in \mathbb{F}_2^5 \,\middle|\, \begin{array}{l} x_1 + x_2 + x_3 = 0 \pmod 2 \\ x_2 + x_4 + x_5 = 0 \pmod 2 \end{array} \right\}. \end{aligned}$$

Graphical Representation of a Code

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$



$$\mathbb{C} = \mathbb{C}_1 \cap \mathbb{C}_2$$

Another example

for graphical representation of a code

Graphical representation

of codeword indicator function

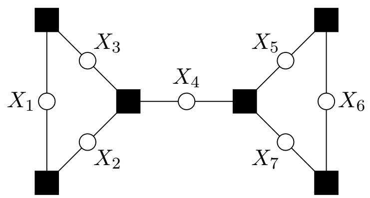
and pseudo-codeword indicator function

Graphical Representation of a Code

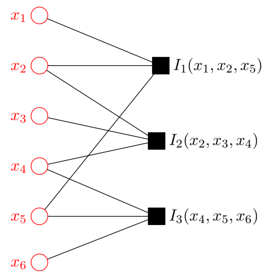
Consider the binary linear code $\mathbb{C}$ defined by the parity-check matrix

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}.$$

This code is also defined by the following Tanner graph:



Factor graph



$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

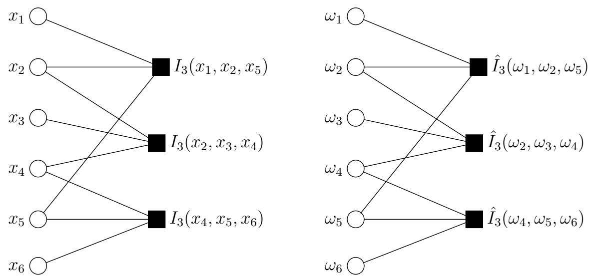
Codeword indicator function:

$$\begin{aligned} I_1(x_1, x_2, x_5) \cdot I_2(x_2, x_3, x_4) \cdot I_3(x_4, x_5, x_6) \\ = \left[(x_1, x_2, x_5) \in \mathbb{C}'_1\right] \cdot \\ \left[(x_2, x_3, x_4) \in \mathbb{C}'_2\right] \cdot \\ \left[(x_4, x_5, x_6) \in \mathbb{C}'_3\right] \end{aligned}$$

Notation: $\mathbb{C}'_1$ is $\mathbb{C}_1$ punctured at all positions except at positions 1, 2, and 5. $\mathbb{C}'_2$ and $\mathbb{C}'_3$ are similarly defined.

Note: $x_i \in \{0, 1\}$

Pseudo-Codewords / Fundamental Polytope



Codeword indicator function:

$$I_1(x_1, x_2, x_5) \cdot I_2(x_2, x_3, x_4) \cdot I_3(x_4, x_5, x_6) = \left[(x_1, x_2, x_5) \in \mathbb{C}'_1\right] \cdot \left[(x_2, x_3, x_4) \in \mathbb{C}'_2\right] \cdot \left[(x_4, x_5, x_6) \in \mathbb{C}'_3\right]$$

Note: $x_i \in \{0, 1\}$

Pseudo-codeword indicator function:

$$\hat{I}_1(\omega_1, \omega_2, \omega_5) \cdot \hat{I}_2(\omega_2, \omega_3, \omega_4) \cdot \hat{I}_3(\omega_4, \omega_5, \omega_6) = \left[(\omega_1, \omega_2, \omega_5) \in \text{conv}(\mathbb{C}'_1)\right] \cdot \left[(\omega_2, \omega_3, \omega_4) \in \text{conv}(\mathbb{C}'_2)\right] \cdot \left[(\omega_4, \omega_5, \omega_6) \in \text{conv}(\mathbb{C}'_3)\right]$$

Note: $0 \le \omega_i \le 1$

Pseudo-Codewords / Fundamental Cone

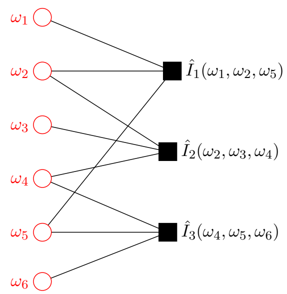
E.g.,

$$\left[(\omega_1, \omega_2, \omega_5) \in \text{conic}(\mathbb{C}'_1)\right] = 1$$

if and only if

$$\omega_1 \le \omega_2 + \omega_5$$
$$\omega_2 \le \omega_1 + \omega_5$$
$$\omega_5 \le \omega_1 + \omega_2$$

$$\omega_1 \ge 0$$
$$\omega_2 \ge 0$$
$$\omega_3 \ge 0$$

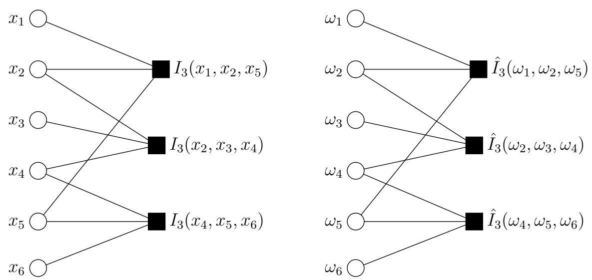


Pseudo-codeword indicator function:

$$\hat{I}_1(\omega_1, \omega_2, \omega_5) \cdot \hat{I}_2(\omega_2, \omega_3, \omega_4) \cdot \hat{I}_3(\omega_4, \omega_5, \omega_6) = \left[(\omega_1, \omega_2, \omega_5) \in \text{conic}(\mathbb{C}'_1)\right] \cdot \left[(\omega_2, \omega_3, \omega_4) \in \text{conic}(\mathbb{C}'_2)\right] \cdot \left[(\omega_4, \omega_5, \omega_6) \in \text{conic}(\mathbb{C}'_3)\right]$$

Note: $0 \le \omega_i$

Pseudo-Codewords / Fundamental Cone



Codeword indicator function:

$$I_1(x_1, x_2, x_5) \cdot I_2(x_2, x_3, x_4) \cdot I_3(x_4, x_5, x_6) = \left[(x_1, x_2, x_5) \in \mathbb{C}'_1\right] \cdot \left[(x_2, x_3, x_4) \in \mathbb{C}'_2\right] \cdot \left[(x_4, x_5, x_6) \in \mathbb{C}'_3\right]$$

Note: $x_i \in \{0, 1\}$

Pseudo-codeword indicator function:

$$\hat{I}_1(\omega_1, \omega_2, \omega_5) \cdot \hat{I}_2(\omega_2, \omega_3, \omega_4) \cdot \hat{I}_3(\omega_4, \omega_5, \omega_6) = \left[(\omega_1, \omega_2, \omega_5) \in \text{conic}(\mathbb{C}'_1)\right] \cdot \left[(\omega_2, \omega_3, \omega_4) \in \text{conic}(\mathbb{C}'_2)\right] \cdot \left[(\omega_4, \omega_5, \omega_6) \in \text{conic}(\mathbb{C}'_3)\right]$$

Note: $0 \le \omega_i$

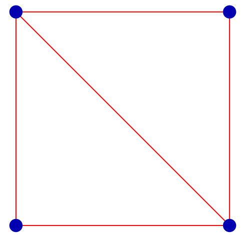
Part 6

Graph-cover decoding

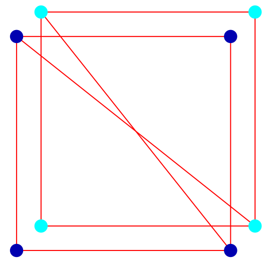
Outline of Part 6

- Definition of graph covers
- Graph-cover (GC) decoding
- Equivalence of GC decoding and LP decoding

Graph Covers



original graph



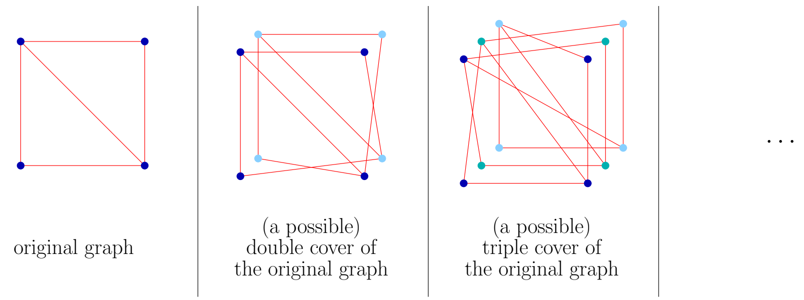
2-fold cover of original graph

Definition: A double cover of a graph is . . .

Note: the above graph has $2! \cdot 2! \cdot 2! \cdot 2! \cdot 2! = (2!)^5$ double covers.

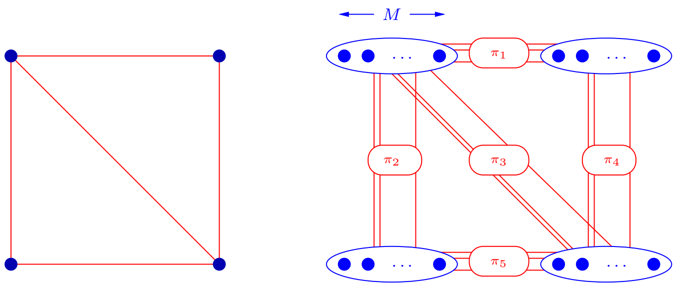
Definition of graph covers

Graph Covers



Besides double covers, a graph has also triple covers, quadruple covers, quintuple covers, etc.

Graph Covers



original graph

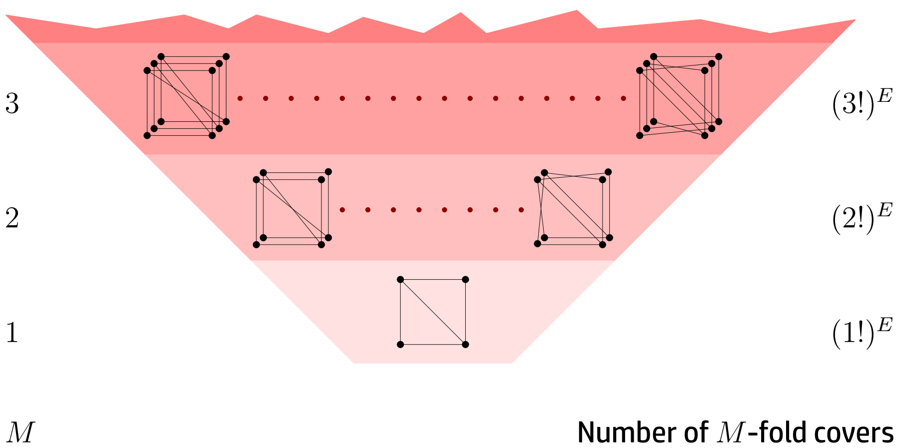
(possible) M-fold cover of original graph

An M-fold cover is also called a cover of degree M. Do not confuse this degree with the degree of a vertex!

Note: a graph G with E edges has $(M!)^E$ M-fold covers.

Graph-cover decoding

Graph Cover Hierarchy



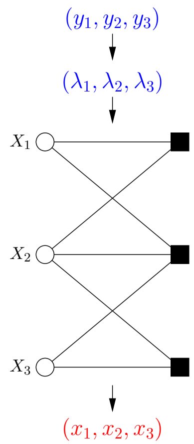
Towards Blockwise Graph-Cover Decoding

Remember, blockwise ML decoding can be formulated as follows:

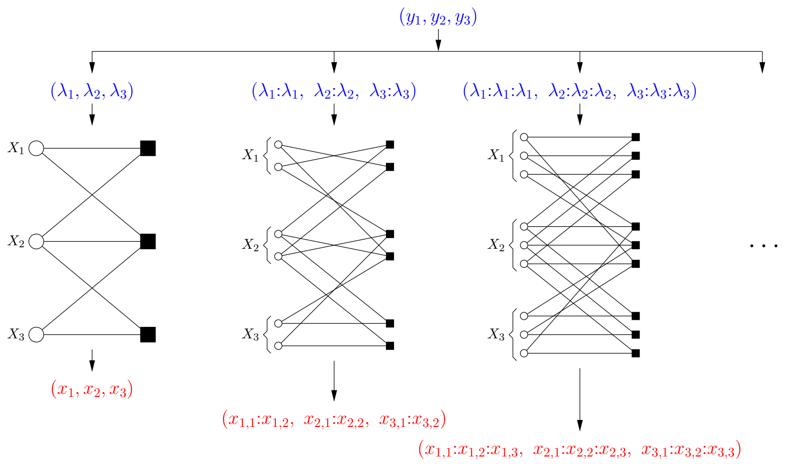
- Receive $\mathbf{y}$.
- Compute the LLR vector $\boldsymbol{\lambda}$.
- Compute

$$\hat{\mathbf{x}}^{\text{block}}_{\text{MAP}}(\mathbf{y}) \triangleq \arg\min_{\mathbf{x}\in\mathcal{C}} \langle \boldsymbol{\lambda}, \mathbf{x} \rangle,$$

where $\langle \cdot , \cdot \rangle$ denotes the standard inner product.



Blockwise Graph-Cover Decoding



Blockwise Graph-Cover Decoding

We now define the graph-cover decoder like this:

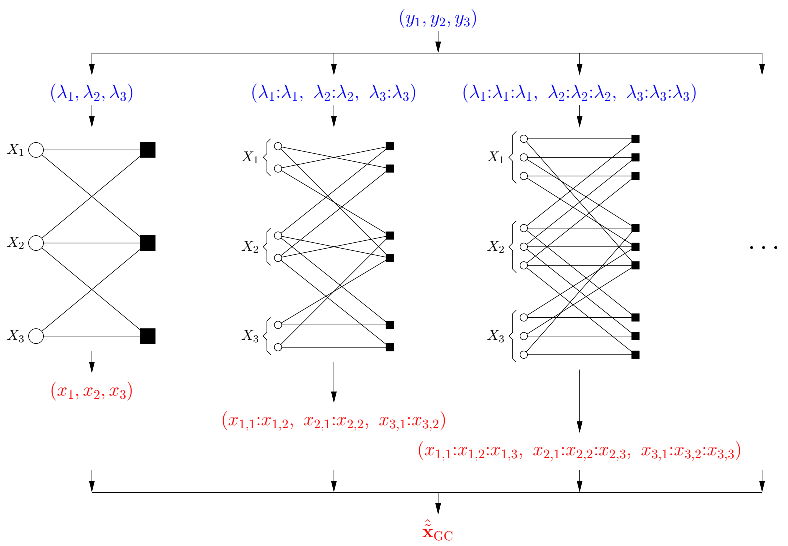
- Receive $\mathbf{y}$.
- Compute the LLR vector $\boldsymbol{\lambda}$.
- Let $\hat{\tilde{\mathbf{x}}}_{\mathrm{GC}}(\mathbf{y})$ be the vector $\tilde{\mathbf{x}}$ of the pair $(\tilde{T}, \tilde{\mathbf{x}})$ that minimizes

$$\min_{(\tilde{T},\tilde{\mathbf{x}}):\ \tilde{T} \text{ is a finite cover of } T(\mathbf{H}),\ \tilde{\mathbf{x}} \in \mathbb{C}(\tilde{T})} \frac{1}{\deg(\tilde{T})} \langle \tilde{\boldsymbol{\lambda}}, \tilde{\mathbf{x}} \rangle.$$

Here we used the following notation:

- $\tilde{\boldsymbol{\lambda}}$ is the lifting of $\boldsymbol{\lambda}$ to $\tilde{T}$.
- $\mathbb{C}(\tilde{T})$ is the code defined by the Tanner graph $\tilde{T}$.
- $\deg(\tilde{T})$ is the degree of the cover $\tilde{T}$ over $T(\mathbf{H})$.

Blockwise Graph-Cover Decoding



Equivalence of GC decoding and LP decoding

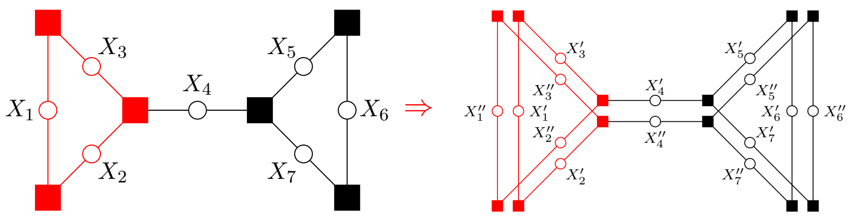
Blockwise Graph-Cover Decoding

What is the connection to LP decoding?

We start by studying codewords in graph covers of some Tanner graph.

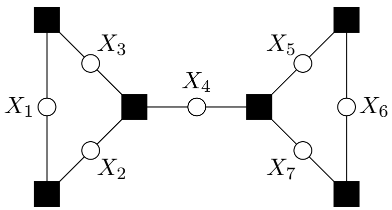
Codewords in Graph Covers

Obviously, any codeword in the base normal factor graph can be lifted to a codeword in the double cover of the base normal graph.

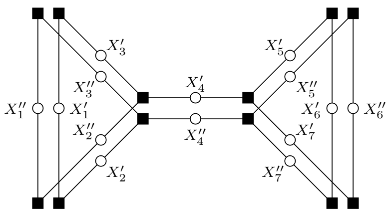


$(1, 1, 1, 0, 0, 0, 0)$ $\quad$ $(1{:}1,\ 1{:}1,\ 1{:}1,\ 0{:}0,\ 0{:}0,\ 0{:}0,\ 0{:}0)$

Codewords in Graph Covers



Base Tanner graph
of a length-7 code

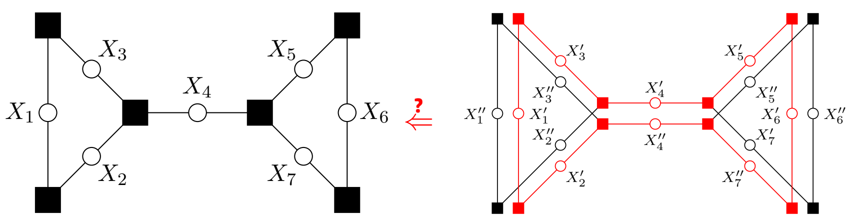


Possible double cover of
the base factor graph

Let us study the codes defined by the graph covers of this base Tanner/factor graph.

Codewords in Graph Covers

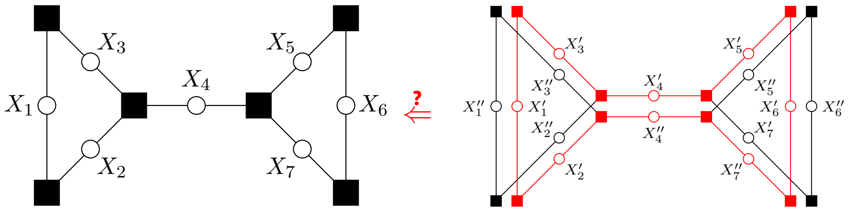
But in the double cover of the base normal factor graph there are also codewords that are not liftings of codewords in the base factor graph!



? $\quad$ $(1{:}0,\ 1{:}0,\ 1{:}0,\ 1{:}1,\ 1{:}0,\ 1{:}0,\ 0{:}1)$

Codewords in Graph Covers

But in the double cover of the base normal factor graph there are also codewords that are not liftings of codewords in the base factor graph!



What about

$\left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{2}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)$?

$(1{:}0,\ 1{:}0,\ 1{:}0,\ 1{:}1,\ 1{:}0,\ 1{:}0,\ 0{:}1)$

Codewords in Graph Covers

Theorem:

- Let $\mathcal{P} \triangleq \mathcal{P}(\mathbf{H})$ be the fundamental polytope of a parity-check matrix $\mathbf{H}$.
- Let $\mathcal{P}'$ be the set of all pseudo-codewords obtained through codewords in finite covers.

Then, $\mathcal{P}'$ is dense in $\mathcal{P}$, i.e.

$$\mathcal{P}' = \mathcal{P} \cap \mathbb{Q}^n$$

$$\mathcal{P} = \text{closure}(\mathcal{P}').$$

Moreover, note that all vertices of $\mathcal{P}$ are vectors with rational entries and are therefore also in $\mathcal{P}'$.

Codewords in Graph Covers

More formally, the

pseudo-codeword $\boldsymbol{\omega} = (\omega_1, \ldots, \omega_n) \in \mathbb{R}^n$

associated with a

valid configuration $\tilde{\mathbf{x}}$ in some M-fold cover $\tilde{\mathsf{G}}$

is defined to be the vector

$$\boldsymbol{\omega} \triangleq \boldsymbol{\varphi}_M(\tilde{\mathsf{G}}, \tilde{\mathbf{x}}) \qquad \text{with} \qquad \omega_i \triangleq \frac{1}{M} \sum_{m=1}^{M} \tilde{x}_{i,m}.$$

Blockwise Graph-Cover Decoding

What is the connection to LP decoding?

Let $\boldsymbol{\omega} \triangleq \boldsymbol{\omega}(\tilde{\mathbf{x}}) \in \mathbb{R}^n$ be the pseudo-codeword associated with $\tilde{\mathbf{x}}$, i.e.,

$$\omega_i(\tilde{\mathbf{x}}) = \frac{1}{\deg(\tilde{T})} \sum_{\ell=1}^{\deg(\tilde{T})} \tilde{x}_{i,\ell}.$$

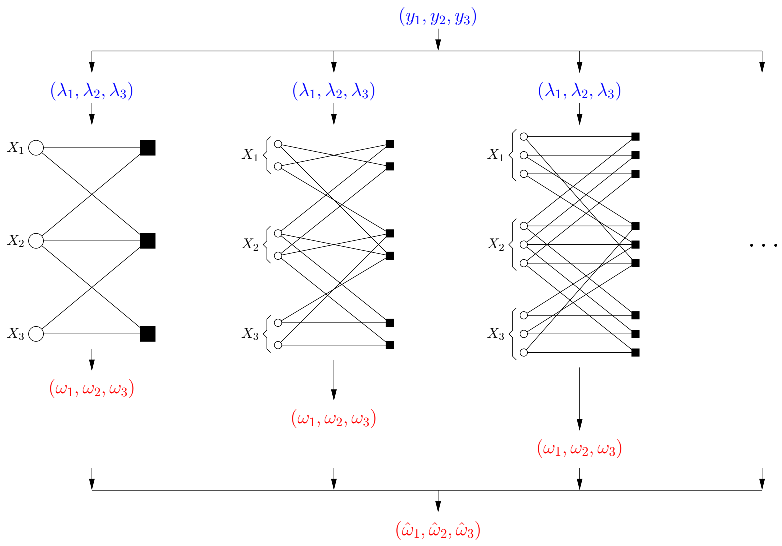
This helps in reformulating the above cost function:

$$\frac{1}{\deg(\tilde{T})} \langle \tilde{\boldsymbol{\lambda}}, \tilde{\mathbf{x}} \rangle = \langle \boldsymbol{\lambda}, \boldsymbol{\omega}(\tilde{\mathbf{x}}) \rangle.$$

Derivation: $\frac{1}{\deg(\tilde{T})} \langle \tilde{\boldsymbol{\lambda}}, \tilde{\mathbf{x}} \rangle = \frac{1}{\deg(\tilde{T})} \sum_{i=1}^{n} \sum_{\ell=1}^{\deg(\tilde{T})} \tilde{\lambda}_{i,\ell} \tilde{x}_{i,\ell} = \sum_{i=1}^{n} \lambda_i \frac{1}{\deg(\tilde{T})} \sum_{\ell=1}^{\deg(\tilde{T})} \tilde{x}_{i,\ell}$

$= \langle \boldsymbol{\lambda}, \boldsymbol{\omega}(\tilde{\mathbf{x}}) \rangle.$

Blockwise Graph-Cover Decoding



Blockwise Graph-Cover Decoding

Using the above observation we can reformulate the minimization problem solved by the blockwise graph-cover decoder to read

$$\begin{aligned} & \min_{(\tilde{T}, \tilde{\mathbf{x}}):\ \tilde{T} \text{ is a finite cover of } T(\mathbf{H}),\ \tilde{\mathbf{x}} \in \mathbb{C}(\tilde{T})} \frac{1}{\deg(\tilde{T})} \langle \tilde{\boldsymbol{\lambda}}, \tilde{\mathbf{x}} \rangle \\ = & \min_{(\tilde{T}, \tilde{\mathbf{x}}):\ \tilde{T} \text{ is a finite cover of } T(\mathbf{H}),\ \tilde{\mathbf{x}} \in \mathbb{C}(\tilde{T})} \langle \boldsymbol{\lambda}, \boldsymbol{\omega}(\tilde{\mathbf{x}}) \rangle \\ = & \min_{\boldsymbol{\omega} \in \mathcal{P}'(T(\mathbf{H}))} \langle \boldsymbol{\lambda}, \boldsymbol{\omega} \rangle \\ = & \min_{\boldsymbol{\omega} \in \mathcal{P}(T(\mathbf{H}))} \langle \boldsymbol{\lambda}, \boldsymbol{\omega} \rangle. \end{aligned}$$

However, the last line is equivalent to the minimization problem solved by the LP decoder!

Blockwise Graph-Cover Decoding

Next, we have to understand the following set:

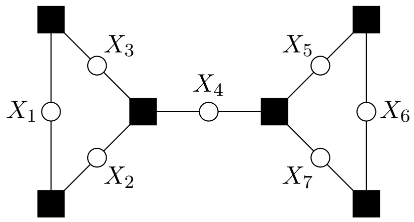
$$\mathcal{P}'(T) \triangleq \left\{ \boldsymbol{\omega}(\tilde{\mathbf{x}}) \in \mathbb{R}^n \;\middle|\; \tilde{\mathbf{x}} \in \mathbb{C}(\tilde{T}),\ \text{where } \tilde{T} \text{ is some finite cover of } T \right\}.$$

However, as we saw before:

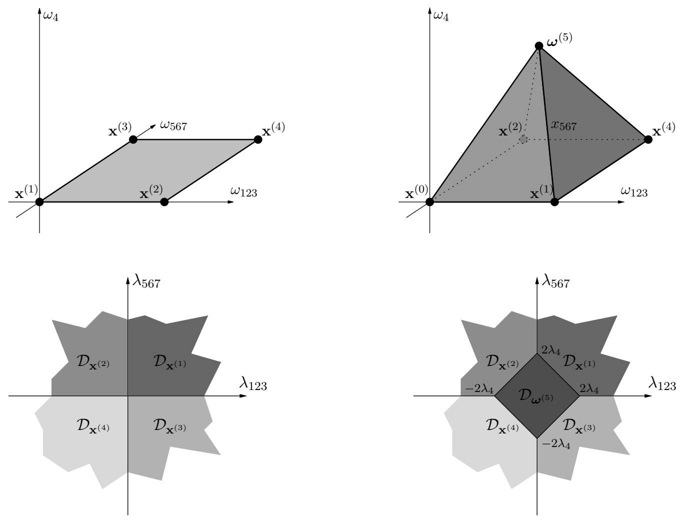
$$\mathcal{P}' = \mathcal{P} \cap \mathbb{Q}^n.$$

Fundamental Polytope / Decision Regions

Consider again the following length-7 code:



Fundamental Polytope / Decision Regions



References

[1] R. Koetter and P. O. Vontobel, "Graph covers and iterative decoding of finite-length codes," Proc. 3rd Intern. Symp. on Turbo Codes and Related Topics, Brest, France, pp. 75–82, Sep. 1–5, 2003.

[2] P. O. Vontobel and R. Koetter, "Graph-cover decoding and finite-length analysis of message-passing iterative decoding of LDPC codes," `http://www.arxiv.org/abs/cs.IT/0512078`, Dec. 2005.

[3] P. O. Vontobel, "Counting in graph covers: a combinatorial characterization of the Bethe entropy function," IEEE Trans. Inf. Theory, vol. 59, no. 9, pp. 6018–6048, Sep. 2013.

Note:

- What is called "graph-cover decoding" in [2] is called "blockwise graph-cover decoding" in [3], in order to distinguish it from "symbol-wise graph-cover decoding" that is also introduced in [3].
- A slight technical difference between [2] and [3] is that in [2] the minimum in the definition of (blockwise) graph-cover decoding is over all finite covers, whereas in [3] the minimum in the definition of blockwise graph-cover decoding is over all finite M-covers, whereby $M \to \infty$.

Final Comment

In the same way that GC decoding gives an alternative view of LP decoding, graph-cover interpretations of other relaxed linear programs can be given.